

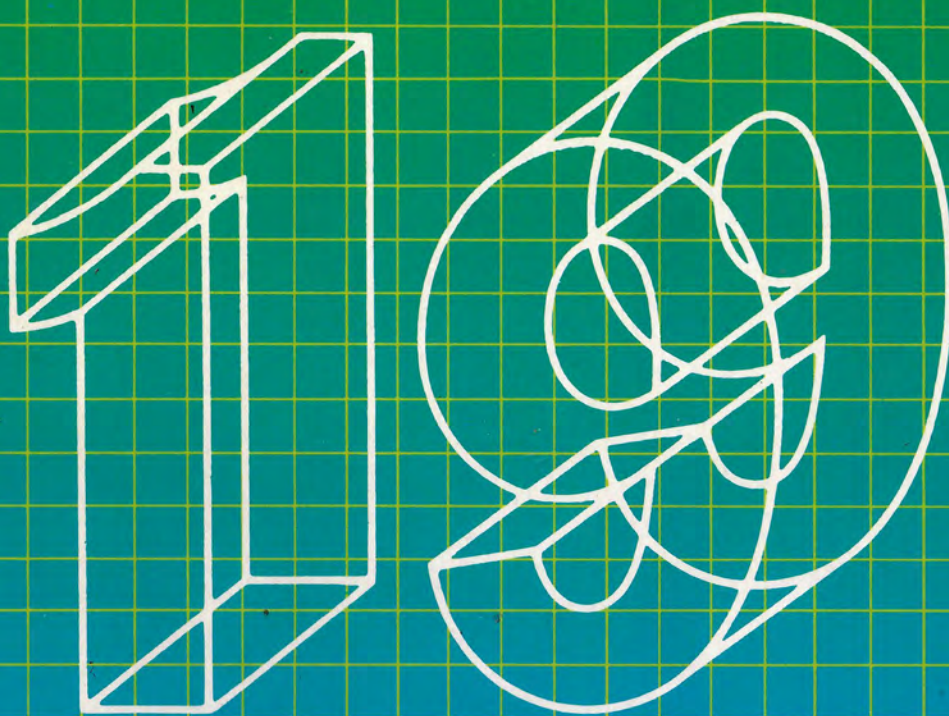
BIBLIOTECA PRACTICA

TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD,

IBM, SPECTRUM, COMMODORE Y MSX



BRICOLAGE INFORMATICO

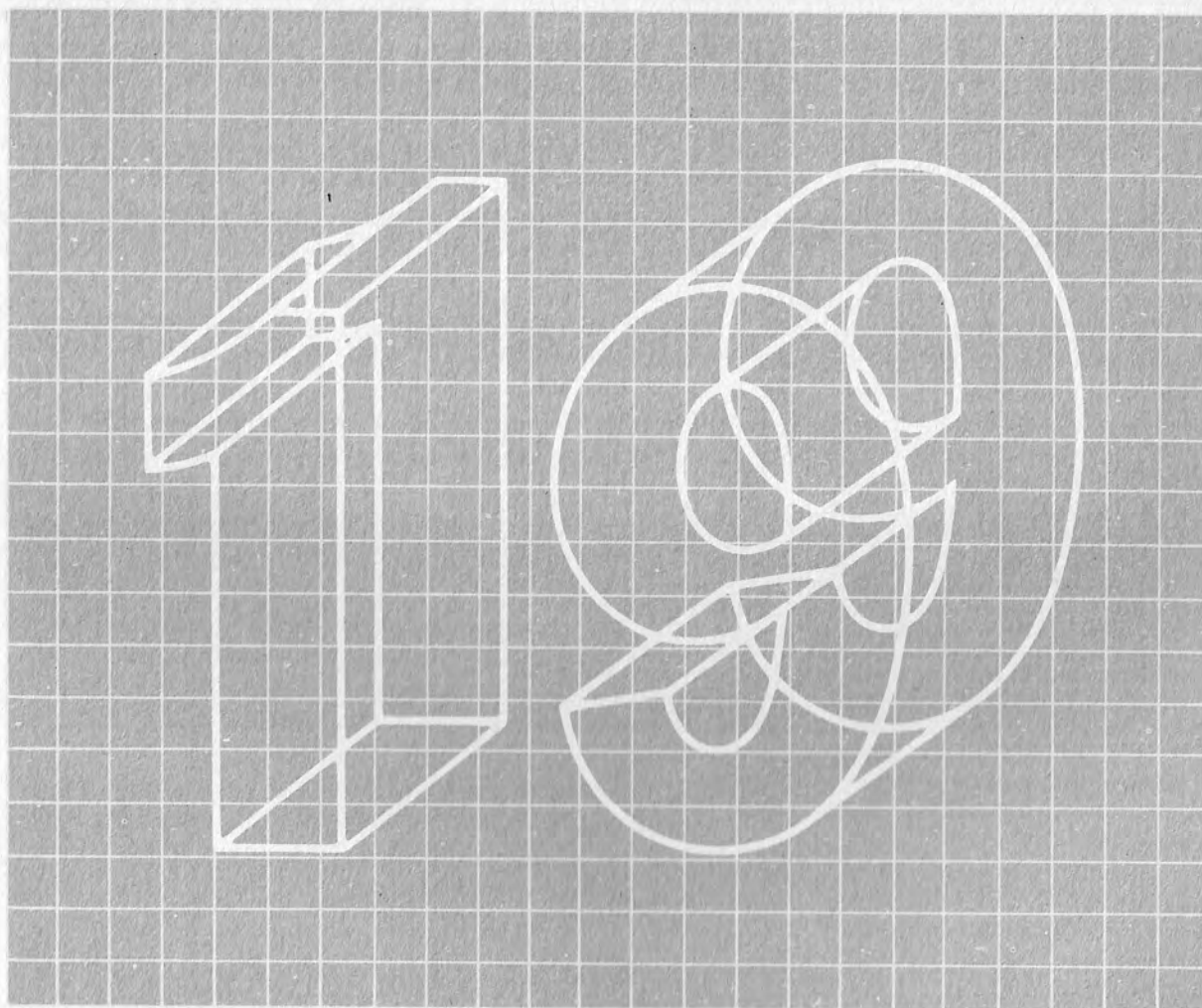
MANEJO DE ELEMENTOS GRAFICOS

EDICIONES SIGLO CULTURAL

BIBLIOTECA PRACTICA TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD,
IBM, SPECTRUM, COMMODORE Y MSX



EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S. A.

Director-editor:

RICARDO ESPAÑOL CRESPO.

Gerente:

ANTONIO G. CUERPO.

Directora de producción:

MARIA LUISA SUAREZ PEREZ.

Director de la colección:

JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño:

BRAVO-LOFISH.

Maquetación:

D. S. M.

Dibujos:

JOSE OCHOA

Tomo 19. «Experiencias y prácticas en logo», Carlos Albert, Técnico de Informática; Adoración Llena Jubero, Técnico de Informática. «Manejo de sprites y elementos gráficos», «Trucos y rutinas básicas», Francisco Morales. Técnico de Informática. «Aprende con el ordenador», AULA DE INFORMATICA APLICADA: Soledad Tamariz-Martel, Diplomada en Telecomunicación; Francisco Blanca, Diplomado en Telecomunicación; Alejandro Marcos, Licenciado en Química; Fernando Suero, Diplomado en Telecomunicación. «El Taller del Hardware», Carlos Rey, Ingeniero Industrial. «Pequeña historia de la Informática», «Temas monográficos de vanguardia», «Terminología», «Vocabulario de Informática», Blanca Arbizu, Técnico de Informática.

Ediciones Siglo Cultural, S. A.

Dirección, redacción y administración:

Pedro Teixeira, 8-2.º. Teléf. 810 52 13. 28020 Madrid.

Publicidad:

Gofar Publicidad, S. A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:

COEDIS, S. A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.

Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:

CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.

Buenos Aires - 1.290. Argentina.

Todos los derechos reservados. Este libro no puede ser, en parte o totalmente, reproducido, memorizado en sistemas de archivo, o transmitido en cualquier forma o medio, electrónico, mecánico, fotocopia o cualquier otro, sin la previa autorización del editor.

ISBN del tomo: 84-7688-111-8

ISBN de la obra: 84-7688-047-2

Fotocomposición:

ARTECOMP, S. A. Albarracín, 50. 28037 Madrid.

Imprime:

MATEU CROMO. PINTO (Madrid).

© Ediciones Siglo Cultural, S. A., 1986

Depósito legal: M-43.185-1986

Printed in Spain - Impreso en España.

Suscripciones y números atrasados:

Ediciones Siglo Cultural, S. A.

Pedro Teixeira, 8-2.º. Teléf. 810 52 13. 28020 Madrid.

Junio, 1987

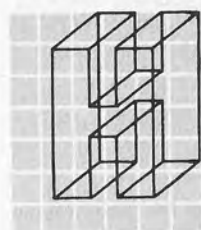
P.V.P. Canarias: 355.—Ptas.

INDICE

■ EXPERIENCIA Y PRACTICAS EN LOGO	5
■ MANEJO DE SPRITES Y ELEMENTOS GRAFICOS	15
■ TRUCOS Y RUTINAS BASICAS	22
■ EL TALLER DEL HARDWARE	33
■ APRENDER CON EL ORDENADOR	39
■ PEQUEÑA HISTORIA DE LA INFORMATICA	48
■ TEMAS MONOGRAFICOS DE VANGUARDIA	53
■ TERMINOLOGIA DE VIDEOTEXTOS	58
■ VOCABULARIO DE INFORMATICA	60

EXPERIENCIA Y PRACTICAS EN LOGO

■ Condiciones



ASTA ahora siempre que hemos establecido una condición dentro de un procedimiento hemos utilizado una relación en la que han intervenido una variable y una constante. La relación establecida entre ambas ha

consistido en una comparación.

Para realizar esta comparación hemos utilizado los operadores de relación:

- = igual que
- > mayor que
- < menor que

y hemos conseguido relaciones como:

- :V = 10 El valor de la variable :V es igual a 10
- :V > 10 El valor de la variable :V es mayor que 10
- :V < 10 El valor de la variable :V es menor que 10

Como resultado de estas relaciones obtenemos CIERTO o FALSO dependiendo que se cumpla o no la condición.

Estas relaciones pueden combinarse a la vez con los operadores lógicos:

- NO
- Y
- O

Para ello hay que expresar la relación entre paréntesis y utilizarla como predicado del operador:

- Y (:V > 10) (:B = 20)

El valor de la variable :V es mayor que 10 y el valor de la variable :B es igual a 20.

- O (:V < 5) (:B > 10)

El valor de la variable :V es menor que 5 o el valor de la variable :B es mayor que 10.

- NO (:V < 5)

No es menor el valor de la variable :V que 5.

Por ejemplo, si la condición que se quiere establecer es que una variable sea mayor o igual a un número, tenemos que expresarlo:

?SI O (:V > 10) (:V = 10)

Obtenemos como resultado CIERTO, siempre que :V contenga un valor igual o superior a diez.

Si la condición es que el valor de una variable sea distinto de diez, lo expresamos de la siguiente forma:

?SI NO (:V = 10)

NO IGUAL equivale a distinto.

Si, por ejemplo, queremos realizar una determinada acción cuando el valor de una variable :A sea igual a siete y el valor de una variable :B sea mayor que 7, podemos expresar de dos formas:

1.ª forma:

?SI :A = 7 [SI :B > [ACCION 1]] [ACCION]

2.ª forma:

?SI Y (:A = 7) (:B > 7) [ACCION 1] [ACCION 2]

Combinando estos dos tipos de operadores podemos ampliar las condiciones y ahorrarnos el tener que utilizar más de las necesarias.

■ La orden número

NUMERO? objeto

Si el objeto LOGO es un número, la or-

Es posible saber de qué tipo es el contenido de una variable.

EXPERIENCIA Y PRACTICAS EN LOGO

den NUMERO? devolverá CIERTO; si no, devolverá FALSO.

```
? ESCRIBE NUMERO? 3
CIERTO
```

El objeto en este caso es el número 3; por tanto, el resultado de NUMERO? es CIERTO.

```
? HAZ "A "G
? ESCRIBE NUMERO? :A
FALSO
```

El objeto es la variable A, el valor de A es la letra G, el resultado de NUMERO? es FALSO.

En muchas ocasiones nos encontramos con la necesidad de saber qué tipo de dato contiene una variable, o de qué tipo es el dato que se ha leído del teclado; a partir de ahora NUMERO? nos informará de ello.

Veamos un ejemplo:

```
? PARA SUMAD
> HAZ "A LEECAR
> SI NUMERO? :A [] [ALTO]
> HAZ "B LEECAR
> SI NUMERO? :B [] [ALTO]
> HAZ "C :A + :B
> ESCRIBE "SUMA
> ESCRIBE :C
> FIN
```

El procedimiento SUMAD lee los datos del teclado; si uno de ellos no es un número, la ejecución del procedimiento finaliza, y si los datos leídos son numéricos, se suman.

Esto que hemos hecho es una forma de impedir que lleguen letras como operandos a la operación suma.

A la hora de leer un carácter del teclado, nos encontramos con que no todas las letras equivalen a un solo carácter.

Hay teclas, como son las de movimiento del cursor, que no pueden representarse con códigos ASCII normales, y por ello al ser leídas retornan un código ASCII ampliado.

Así en LOGO para leer una tecla de este tipo necesitamos leer dos caracteres, lo cual no significa que se deban pulsar dos teclas; estas teclas especiales ya suponen en sí dos caracteres.

Hay teclas que no pueden representarse en códigos ASCII normales, por ello, al ser leídas retornan un código ASCII ampliado.

El ASCII del primer carácter será cero. El ASCII del segundo carácter será el que en la tabla de códigos ASCII ampliados le corresponda a la tecla que hayamos pulsado.

Compruébalo con este ejemplo:

```
? PARA COD
> HAZ "A LEECAR
> ESCRIBE "PRIMERO
> ESCRIBE ASCII :A
> HAZ "B LEECAR
> ESCRIBE "SEGUNDO
> ESCRIBE ASCII :B
> FIN
```

Ejecuta el procedimiento.

```
? COD
```

y pulsa, por ejemplo, la tecla de movimiento de cursor hacia arriba.

En la pantalla aparecerá:

```
PRIMERO
O
CIERTO
SEGUNDO
72
```

A continuación presentamos un ejemplo en el que la utilización de las órdenes SI y LEECAR es continua e imprescindible.

— DEFINIR CARACTERES DE 8 * 8.

Definimos un carácter iluminando puntos en una cuadrícula de 8 por 8 y a su vez vamos dibujando, a la izquierda de la cuadrícula, el carácter que estamos definiendo en tamaño real.

Las normas que seguimos son:

— Movemos la tortuga por la cuadrícula con las teclas de movimiento de cursor.

— Para iluminar un punto pulsamos la tecla RETURN, cuando la tortuga esté en la cuadrícula correspondiente.

— Para finalizar la definición de carácter pulsamos la letra F.

— La cuadrícula está formada por cuadros de 15 por 15; por tanto, cuando movemos la tortuga una posición saltamos 15 puntos, tanto para la X como para la Y.

— Trabajamos con letras mayúsculas.

```

? PARA DEFI
> BP
> OT
> BL
> REPITE 4 [AV 120 GD 90]
> REPITE 7 [GD 90 AV 15 GI 90 AV 120
RE 120]
> CENTRO
> REPITE 7 [AV 15 GD 90 AV 120 RE
120 GI 90] SL
> PONPOS [52 63]
> HAZ "X 52
> HAZ "Y 63
> HAZ "PX -20
> HAZ "PY8
> MT
> REPITE 1000 [POSI SI (ASCII :B) = 70
[ALTO]]
> FIN

```

Este primer procedimiento se encarga de:

- Inicializar la pantalla.
- Dibujar una cuadrícula de 8 por 8.
- Posicionar la tortuga dentro de la cuadrícula.

La variable Y contiene la posición (coordenada Y) de la tortuga dentro de la cuadrícula.

La variable X contiene la posición (coordenada X) de la tortuga dentro de la cuadrícula.

Dibujo real del carácter

La variable PY contiene la posición (coordenada Y), en donde se pintará un punto si el equivalente en la cuadrícula se ilumina.

La variable PX contiene la posición (coordenada X) del punto del dibujo real.

- Ejecutar, si el código ASCII del contenido de la variable B (esta variable toma valor en el procedimiento POSI) no es 70 (letra F), el procedimiento POSI 1000 veces.

Utilizamos el REPITE 1000 [] para conseguir ejecutar el procedimiento POSI el número de veces suficientes para definir un carácter.

Si conociésemos una orden del tipo:

MIENTRAS se cumpla la condición ejecuta POSI no tendríamos que usar un método tan rebuscado como éste; más adelante vere-

mos cómo conseguimos simular una orden MIENTRAS.

Segundo procedimiento (procedimiento POSI):

```

? PARA POSI
> HAZ "B LEECAR
> SI (ASCII :B) = 13 [BL RELLENA PIN-
TA HAZ "B LEECAR]
> SI (ASCII :B) - 70 [ALTO]
> HAZ "A LEECAR
> HAZ "C ASCII :A
> SI :C = 72 [HAZ "Y :Y + 15 HAZ "PY
:PY + 1]
> SI :C = 80 [HAZ "Y :Y - 15 HAZ "PY
:PY - 1]
> SI :C = 75 [HAZ "X :X - 15 HAZ "PX
:PX - 1]
> SI :C = 77 [HAZ "X :X + 15 HAZ "PX
:PX + 1]
> PONX :X
> PONY :Y
> FIN

```

Las funciones del procedimiento POSI son:

- Leer un carácter del teclado.
- Analizar el carácter leído.

Si el ASCII del carácter es 13 (significa que se ha pulsado la tecla RETURN, iluminar un punto), se ejecuta el procedimiento PINTA y se lee otro carácter.

Si el ASCII del carácter es 70, se devuelve el control al procedimiento DEFI, que a su vez finaliza.

- Leer otro carácter (para leer las teclas del cursor necesitamos dos LEECAR).
- Analizar el carácter leído.

Si el ASCII del carácter es 72, se ha pulsado la tecla ↑, incrementamos la variable Y en 15 y la variable PY en 1.

Si el ASCII del carácter es 80, se ha pulsado la tecla ↓, decrementamos la variable Y en 15 y la variable PY en uno.

Si el ASCII del carácter es 75, se ha pulsado la tecla ←, decrementamos la variable X en 15 y la variable PX en 1.

Si el ASCII del carácter es 77, se ha pulsado la tecla →, incrementamos la variable X en 15 y la variable PX en 1.

En LOGO, para leer una tecla de código ASCII ampliado, necesitamos leer dos caracteres.

EXPERIENCIA Y PRACTICAS EN LOGO

— Colocar la tortuga en la nueva posición.

? PARA PINTA

> OT

> SL

> PONX :PX

> PONY :PY

> BL

> AV 1

> SL

> PONX :X

> PONY :Y

> MT

> FIN

PINTA es el procedimiento que se ejecuta si se ha pulsado la tecla RETURN.

Este procedimiento oculta la tortuga con el fin de que no se vea el desplazamiento que realiza, pinta un punto en las coordenadas especificadas por las variables PX y PY, devuelve la tortuga a su antigua posición y la visualiza.

Estos tres procedimientos nos sirven para definir un carácter de 8 por 8 a través de una cuadrícula.

Define tus primeros caracteres.

?DEFI

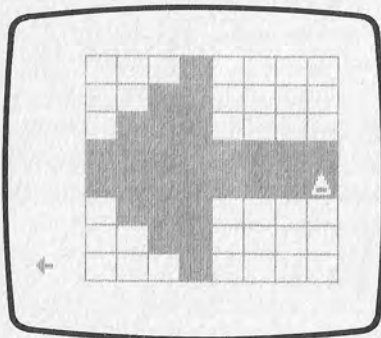


Fig. 1.

? DEFI

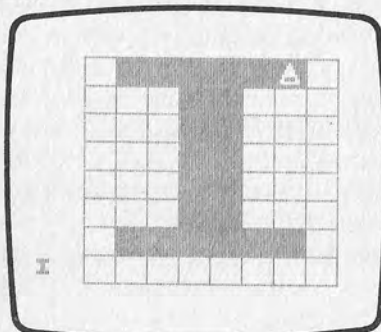


Fig. 2.

Puedes controlar el movimiento de la tortuga desde el teclado.

En el SPECTRUM y en el MSX no existen teclas codificadas en código ASCII ampliado; por tanto, sólo necesitamos una orden (lee-carácter) para leer las teclas de movimiento del cursor.

Los códigos ASCII de las teclas de movimiento del cursor en el SPECTRUM son:

Tecla	Código
↑	11
↓	10
→	9
←	8

En el MSX:

Tecla	Código
↑	30
↓	31
→	28
←	29

Recordamos que en el SPECTRUM la orden

HAZ "nombre objeto

equivale a:

ASIGNA "nombre objeto

■ A los usuarios del Amstrad

Vamos a hacer otro pequeño resumen de las órdenes que posee el Logo del AMSTRAD con relación a la que ya hemos visto para otros ordenadores.

— OPERADORES ARITMETICOS

Al igual que en las demás versiones, el Logo del AMSTRAD dispone de las operaciones:

- SUMA
- RESTA
- MULTIPLICACION
- DIVISION

expresado mediante los operadores:

- +
- -
- *
- /

Podemos obtener el resultado de una operación realizada con estos operadores mediante la orden:

— ESCRIBIR "expresión aritmética

? ESCRIBIR "8 + 2

y obtenemos en la pantalla:

10

? ESCRIBIR "8 - 2

y obtenemos

6

? ESCRIBIR "8 * 2

y obtenemos

16

? ESCRIBIR "8 / 2

y obtenemos

4

No disponemos de los operadores en forma prefija:

— SUMA

— RESTO

— PRODUCTO

— COCIENTE

No posee órdenes que vimos que sí disponían otras versiones de Logo, como son:

— ENTERO n (devuelve la parte entera de n)

— REDONDEA n (devuelve el valor de n redondeado al entero más próximo)

— ABS n (devuelve el valor absoluto de n)

— RAIZCUADRADA (RC) n (devuelve el valor de la raíz cuadrada de n).

Tampoco dispone de las órdenes:

— EXP

— LN

— PI

— LA ORDEN HAZ

La orden HAZ en la versión del Logo para el AMSTRAD, es igual a la explicada para otros ordenadores.

Para asignar un valor a una variable lo hacemos:

? HAZ "V 15

Asignamos el valor 15 a la variable :V

No dispone de la orden ASIGNA.

El Logo del AMSTRAD posee otra orden para asignar desde el exterior valores a las variables. Se trata de:

— ENTRADA "nombre de variable

Si, por ejemplo, introducimos:

? ENTRADA "VALOR

el Logo esperará a que sea introducido algo desde el teclado, y una vez realizado, pasa a ser el contenido de la variable VALOR.

ENTRADA es similar a la orden LEECAR, vista para otros ordenadores.

Para conocer el valor que contiene una variable podemos utilizar dos métodos:

— ? HAZ "V 7

? ESCRIBIR :V

Asignamos un 7 a la variable :V y a continuación ordenamos que se escriba en la pantalla el valor de dicha variable. Obtendremos en la pantalla un 7.

— ? HAZ "V 7

? :V

Basta con expresar el nombre de la variable de la que queremos conocer su valor.

— GESTION CON VARIABLES

De todos los ordenadores que vimos que gestionaban las variables, el Logo del AMSTRAD sólo nos permite el borrar una variable. Esto lo conseguimos con la orden:

? BORRAR :nombre de variable

La variable que se especifica se borrará de la memoria del ordenador.

— OPERADORES DE RELACION

Los operadores de relación que posee son los mismos que hemos visto para otros ordenadores.

— = :igual que

— > :mayor que

— < :menor que

— OPERADORES LOGICOS

Los operadores lógicos de la versión del Logo del AMSTRAD son los mismos que hemos visto para otras versiones. La única di-

Si vas a realizar una operación matemática, asegúrate que los operandos son números y no letras.

EXPERIENCIA Y PRACTICAS EN LOGO

ferencia es que en este caso se tienen que escribir en inglés:

- O equivale a OR
- Y equivale a AND
- NO equivale a NOT

La forma de expresarlos es:

- OR
OR (:V < 10) (:B < 20)
- AND
AND (:V = 10) (:B > 20)
- NOT
NOT (:V = 10)

Se escriben, en primer lugar, el operador y a continuación los o el predicado.

- LA ORDEN SI

La orden SI tiene algunas diferencias, con respecto a su utilización, de otras versiones del Logo que ya hemos visto.

La forma de establecer la condición es:

? SI (condición)

Observa que la condición debe ir entre paréntesis y que no se indica a continuación ninguna acción a tomar.

Sólo se puede indicar la acción a tomar en el caso que la condición se cumpla. Esta acción debe asignarse previamente a una variable llamada "instrucción" mediante la orden HAZ.

? HAZ "instrucción [acción]

Por ejemplo:

- ? HAZ "instrucción [adelante 40]
- ? HAZ "V 30
- ? SI (:V = 30)

Asignamos a la variable instrucción la acción a tomar para el caso que la condición se cumpla, que consiste en que la tortuga avance 40 puntos. A continuación le asignamos a la variable :V el valor 30.

La condición que se establece es que la variable :V sea igual a 30. Como el resultado es CIERTO se realiza la acción que hay en instrucción.

Si la condición no se cumple, continuaría con la ejecución de la siguiente orden.

La orden SI en el AMSTRAD permite también combinar las relaciones con los operadores lógicos que hemos visto anteriormente.

Otras órdenes que hemos estudiado y que tampoco posee esta versión del Logo son:
Las órdenes trigonométricas:

SEN
COS
ARCTAN

Y órdenes como:

ALTO
TECLA?
NUMERO?
ASCII
CAR
COPIADEF

Cuadro resumen

- NUMERO? objeto

Devuelve CIERTO si *objeto* es un número; si no, devuelve FALSO.

Ejercicios

1. Intenta resolver mediante un procedimiento las ecuaciones de segundo grado con raíces reales.

Si las raíces de la ecuación son imaginarias, mostrar el mensaje:

IMAGINARIAS

2. Define un procedimiento que devuelva el código ASCII de cualquier tecla que pulsesmos.
3. Intenta realizar los siguientes dibujos:

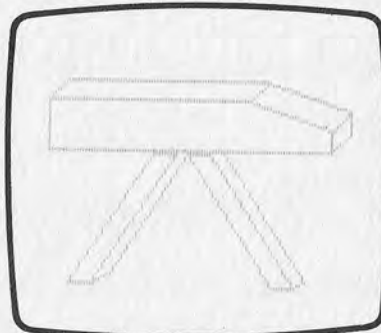


Fig. 3.

Si utilizas operadores lógicos en las condiciones, no te olvides de poner las expresiones entre paréntesis.

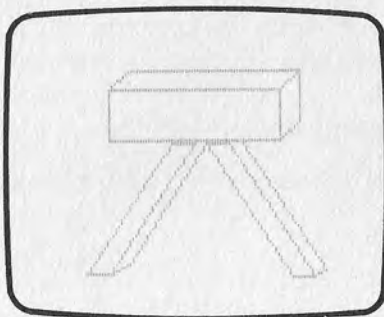


Fig. 4.

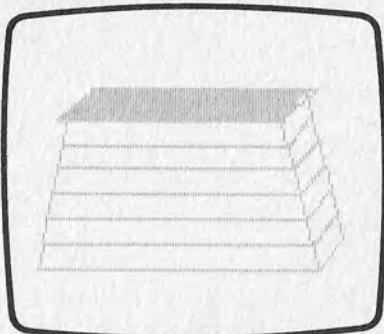


Fig. 5.

4. Define los procedimientos necesarios para que moviendo la tortuga con las teclas de movimiento del cursor podamos dibujar directamente sobre la pantalla lo que queramos.

Además de las teclas del cursor, utiliza las siguientes:

- + para incrementar el número de puntos que avanza o retrocede la tortuga en cada paso.
- acción contraria a la anterior.
- B para bajar el lápiz de la tortuga.
- S para subir el lápiz de la tortuga.
- G para activar la goma.
- F para terminar de dibujar.

5. ¿Son correctas las siguientes órdenes?
- ?HAZ "A "S
?SI NUMERO? 3 / :A [ESCRIBE "CIERTO] [ESCRIBE "FALSO]
 - ?HAZ "A LEECAR
?HAZ "B LEECAR
?SI :A > B REPITE 100 [ESCRIBE :A]
 - ?HAZ "A LEECAR
?SI NUMERO? :A [HAZ "B LEECAR]
[HAZ "C ASCII :A]

- ?HAZ "V AZAR 500
?SI (:V > 300) Y (:V < 350) [ALTO]
- ?HAZ "V 10
?HAZ "B AZAR 10
?SI :V = 10 O :B < 5 [AV 20] [RE 20]

■ Solución a los ejercicios

1:

Con el siguiente procedimiento calculamos las raíces de cualquier ecuación de segundo grado, siempre y cuando las raíces no sean imaginarias.

```
?PARA ECU :A :B :C
> PT
> BT
> HAZ "D -:B
> HAZ "E (:B * :B) - (4 * :A * :C)
> SI :E <= 0 [ESCRIBE "IMAGINARIAS
ALTO]
> HAZ "F RC :E
> HAZ "G 2 * :A
> HAZ "S1 (:D + :F) / :G
> HAZ "S2 (:D - :F) / :G
> ESCRIBE "SOLUCION: 1
> ESCRIBE :S1
> ESCRIBE "SOLUCION: 2
> ESCRIBE :S2
> FIN
```

En matemáticas calculamos las raíces de una ecuación de segundo grado de la siguiente forma:

Sea la ecuación:

$$ax + bx + c = 0$$

las raíces las obtenemos con la expresión:

$$x = \frac{-b \pm \sqrt{b^2 - 4axc}}{2xa}$$

Si el resultado de $(b^2 - 4axc)$ es negativo, las raíces de la ecuación son imaginarias.

— Explicación del procedimiento.

El procedimiento ECU tiene tres entradas:

- A coeficiente de la x
- B coeficiente de la x
- C término independiente.

Dentro del procedimiento encontramos la variable E, que contiene en resultado las expresiones $(B^2 - 4 A C)$.

Combina operadores de relación con lógicos para establecer las condiciones.

EXPERIENCIA Y PRACTICAS EN LOGO

Si el valor de E es negativo, escribimos IMAGINARIAS.

Si el valor de E es mayor o igual que cero, calculamos las dos raíces y las escribimos.

Calcula, por ejemplo, las raíces de la ecuación

$$x^2 + 4x + 3 = 0$$

ECU 1 4 3

el resultado será

SOLUCION: 1
-1

SOLUCION: 2
-3

si las raíces de la ecuación son imaginarias, por ejemplo, las de la ecuación

$$x^2 + 2x + 3 = 0$$

?ECU 1 2 3

el resultado será

IMAGINARIAS

2:

El procedimiento CODIGOS devuelve el código ASCII de cualquier tecla que pulsemos.

```
?PARA CODIGOS
> HAZ "A LEECAR
> HAZ "C ASCII :A
> SI :C = 0 [HAZ "C ASCII LEECAR]
> ESCRIBE :C
> FIN
```

Leemos un carácter de teclado y asignamos a la variable C su código ASCII correspondiente.

Si el valor de C es cero, nos encontramos ante un caso de código ASCII ampliado, leemos otro carácter y almacenamos en la variable C su código ASCII.

Por último, escribimos el valor de la variable.

Para saber el código ASCII de la letra A, teclea:

?CODIGOS

y a continuación la letra A.

En la pantalla aparecerá el número 65.

Para obtener los códigos ASCII de 50 teclas, ejecutamos 50 veces el procedimiento CODIGOS.

?REPITE 50 [CODIGOS]

A continuación vamos pulsando las 50 teclas.

3:

En primer lugar, definimos el procedimiento que dibuja la parte superior del caballo.

? PARA CABALLO

```
> BP
> SL
> OT
> AV 20 GI 90
> RE 45
> BL
> AV 100 GD 90
> AV 20 GD 90
> AV 70 GD 20
> AV 32 GD 70
> AV 9 GI 135
> AV 10 GI 45
> AV 9 GI 70
> AV 32 GI 20
> AV 70 GI 45
> AV 10
> SL
> GI 135 AV 71
> GI 45
> BL
> AV 10
> SL
> GD 65 AV 31
> GD 110
> BL
> AV 10
> SL
> CENTRO
> RE 33 GI 90
> AV 40
> PATAS
> FIN
```

Este procedimiento hace una llamada PATAS para dibujar las patas del caballo.

? PARA PATAS

En una condición se puede introducir más de un operador lógico.

```

> BL
> REPITE 2 [AV 8 GD 120 AV 60 GD 60]
> GD 150 AV 8
> GI 30 AV 55
> PONRUMBO 0
> SL
> RE 52
> GD 90
> AV 30
> BL
> REPITE 2 [AV 8 GI 120 AV 60 GI 60]
> AV 8
> GI 30 AV 4
> GI 90 AV 58
> FIN

```

Una vez tenemos estos dos procedimientos definidos podemos ejecutarlos para ver el dibujo:

? CABALLO

Para dibujar el potro definimos también un procedimiento, en primer lugar, para dibujar la parte superior:

```

? PARA POTRO
> BP
> OT
> SL
> AV 20 GI 90
> RE 40
> BL
> REPITE 2 [AV 60 GD 90 AV 20 GD 90]
> GD 135 AV 10
> GI 45 AV 20
> GI 90 AV 60
> GI 45 AV 10
> GI 135 AV 60
> GI 45 AV 10
> SL
> CENTRO
> RE 33 GI 90
> AV 20
> PATAS
> FIN

```

En POTRO realizamos también una llamada al procedimiento que dibuja las patas. Para ejecutarlo tenemos que introducir:

? POTRO

Por último, definimos los procedimientos que van a dibujar el plinto:

? PARA PLINTO

```

> BP
> OT
> SL
> RE 30 GD 90
> AV 50
> CAJON 100
> CAJON 96
> CAJON 92
> CAJON 88
> CAJON 84
> CAJON 80
> PONRUMBO 270
> REPITE 2 [RE 1 AV 80 GD 135 AV 18 GD 45]
> SL
> GD 45 AV 5
> BL
> RELLENA
> FIN

```

Este procedimiento sitúa la tortuga donde tiene que empezar a dibujar y a continuación hace una serie de llamadas a CAJON dando diferentes valores a la variable que utiliza:

Una vez dibujados los cajones, se dibuja la parte superior del plinto:

El procedimiento que dibuja los cajones es:

```

? PARA CAJON :L
> PONRUMBO 270
> BL
> AV :L
> GD 100
> AV 10 GD 80
> AV :L - 4
> GD 80 AV 10
> GI 135 AV 15
> GI 45 AV 10
> GI 135 AV 15
> FIN

```

La variable :L es la que determina la longitud que va a tener cada uno de los cajones que tiene un plinto.

Para ejecutarlo, basta con introducir:

? PLINTO

4:

La palabra LOGO la hemos dibujado con los siguientes procedimientos.

Asigna el valor introducido desde el teclado a una variable.

EXPERIENCIA Y PRACTICAS EN LOGO

Primer procedimiento

```
? PARA DEFID
> BP
> BL
> MT
> HAZ "P 1
> REPITE 1000 [POSID SI :F = 70
[ALTO]]
> FIN
```

En este primer procedimiento inicializamos la pantalla y ejecutamos 1000 veces, salvo que el valor de la variable F sea 70, el procedimiento POSID, que, como veremos, se encarga de leer una tecla y de analizarla.

Segundo procedimiento

```
? PARA POSID
> HAZ "B LEECAR
> HAZ "F ASCII :B
> SI :F = 43 [HAZ "P :P + 1 ALTO]
> SI :F = 45 [HAZ "P :P - 1 ALTO]
> SI :F = 66 [BL ALTO]
> SI :F = 83 [SL ALTO]
> SI :F = 71 [GOMA ALTO]
> SI :F = 70 [ALTO ALTO]
> HAZ "B LEECAR
> HAZ "F ASCII :B
> SI :F = 72 [PONRUMBO 0 A V :P]
> SI :F = 80 [PONRUMBO 180 AV :P]
> SI :F = 75 [PONRUMBO 270 AV :P]
> SI :F = 77 [PONRUMBO 90 AV :P]
> FIN
```

El procedimiento POSID lee una tecla y ejecuta las acciones que le hemos asignado a dicha tecla.

Recuerda que estamos utilizando letras mayúsculas.

5:

— ? HAZ "A "S

? SI NUMERO? :A [ESCRIBE "CIERTO] [ESCRIBE "FALSO]

CORRECTO

Para el LOGO las órdenes son correctas, aunque el resultado de la orden SI sea FALSO.

— ? HAZ "A LEECAR
? HAZ "B LEECAR
? SI :A > :B [ESCRIBE :A]

Este ejemplo será correcto o incorrecto, dependiendo de los datos que introduzcamos desde el teclado.

Si introducimos dos números, será todo CORRECTO.

Si introducimos alguna letra (por ejemplo la D), el LOGO devolverá el mensaje:

> NO ACEPTA D COMO DATO.

— ? HAZ "A 3
? SI NUMERO? :A REPITE 20 [ESCRIBE "NUMERO]
INCORRECTO

La acción que debe ejecutarse si la condición se cumple no está encerrada entre corchetes.

— ? HAZ "V AZAR 500
? SL Y (:V > 300) (:V < 350) [ALTO]

CORRECTO

La acción se realizará cuando la variable :V contenga un número comprendido entre 301 y 349 ambos inclusive.

— ? HAZ "V 10
? HAZ "B AZAR 10
? SI O :V = 10 :B < 5 [AV 20] [RE 20]

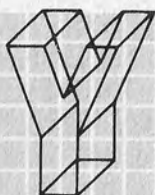
INCORRECTA.

Las dos relaciones que hay en la condición hay que encerrarlas entre paréntesis, ya que estamos utilizando un operador lógico.

El código ASCII de las letras mayúsculas es diferente al de las minúsculas.

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

FANTASMAS EN TU ORDENADOR



A estamos casi en disposición de crear nuestro primer juego completo. Lo único que nos falta es ver algunas cosas que nos hemos dejado sueltas a lo largo de la obra, explicar el color en los ordenadores y dar algunos programas para ayudar al usuario a definirse sus propios caracteres gráficos.

Lo primero que vamos a ver es cómo

realizar tablas de RECORDS. Se recordará que en el tomo 12 vimos el programa del GUSANO LOCO. En este programa se incluía al final una rutina que nos permitía introducir nuestro nombre dentro de una tabla de RECORDS si nuestra puntuación estaba entre las mejores. En su momento, cuando explicamos el funcionamiento del programa, dejamos sin explicar esta rutina en espera de un momento más adecuado para hacerlo. El programa que aparece a continuación es una copia de dicha rutina.

```
2450 REM
2460 REM *****
2470 REM * TABLA DE GANADORES *
2480 REM *****
2490 REM
2500 FOR I=1 TO 1000
2510 NEXT I
2520 CLS:PRINT "PUNTUACIONES DE HOY"
2530 PRINT:PRINT "-----"
2540 FOR I=4 TO 17
2550   LOCATE I,1
2560   PRINT "!";TAB(40);"!"
2570 NEXT I
2580 PRINT "-----"
2590 IF SC>=S(10) THEN LET S(10)=SC:LET A$(10)=CHR$(254)
2600 FOR I=1 TO 10
2610   FOR J=1 TO I
2620     IF S(J)<S(I) THEN LET A=S(J):LET S(J)=S(I):LET S(I)=A:LET B$=A$(J):LET A$(J)=A$(I):LET A$(I)=B$
2630   NEXT J
2640 NEXT I
2650 FOR I=1 TO 10
2660   IF A$(I)=CHR$(254) THEN LET PP=I
2670 NEXT I
2680 IF PP=0 THEN GOTO 2700
2690 A$(PP)="....."
2700 FOR I=1 TO 10
2710   LOCATE 5+I,3
```

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```
2720 PRINT A$(I);TAB(30);S(I)
2730 NEXT I
2740 LOCATE 5+PP,3,1
2750 LET A$(PP)=""
2760 IF PP=0 THEN GOTO 2850
2770 FOR I=1 TO 15
2780 LET A$=INKEY$:IF A$="" THEN GOTO 2780
2790 IF A$=CHR$(8) AND I>1 THEN PRINT CHR$(29);".":CHR$(29);:LET I=I-1:LET A$(PP)
)=MID$(A$(PP),1,I)
2800 IF A$=" " THEN GOTO 2830
2810 IF A$=CHR$(13) THEN LET I=15:GOTO 2840
2820 IF A$<"0" OR A$>"z" THEN GOTO 2780
2830 PRINT A$;:LET A$(PP)=A$(PP)+A$
2840 NEXT I
2850 LOCATE 1,1,0
2860 LOCATE 20,1
2870 PRINT "<<< PULSA UNA TECLA >>>"
2880 LET A$=INKEY$:IF A$="" THEN GOTO 2880
2890 LOCATE 20,1
2900 GOTO 2330
```

No se incluyen las modificaciones que hay que hacerle a este programa para que funcione en ordenadores distintos del IBM, porque ya se vieron en el tomo 12. La única razón de que aparezca esta rutina aquí es para que sirva de referencia a la explicación que de ella vamos a dar.

Antes de empezar con esta rutina hay que decir que hay muchas maneras de realizar una tabla de RECORDS. Este primer programa nos ilustra una de las formas más sencillas de hacerlo, pero hay muchas otras.

Para realizar una rutina que se encargue de gestionar los RECORDS conseguidos por el jugador hay que tener en cuenta que la rutina tiene que tener las siguientes características:

- Ha de haber un vector que almacene los nombres de los jugadores que han conseguido un récord.

- También tiene que haber un vector con los puntuaciones de cada uno de estos jugadores.

- Es conveniente que al principio del programa se almacenen en dichos vectores unos nombres y unas puntuaciones prefijadas por el programador para que la tabla de RECORDS no esté vacía. También es conveniente hacerlo para que el jugador, al ver la tabla, se empeñe más en conseguir una buena puntuación para ganar a un oponente imaginario que consiguió una puntuación más alta que la suya.

- La tabla ha de estar ordenada de mayor a menor puntuación.

Una vez vistas las características que tiene que tener una tabla de puntuaciones, vamos a ver algunos tipos de éstas.

Hay muchas formas de realizar tablas de puntuación. Cada programa comercial realiza su propio tipo de tabla, pero podemos decir que todas se engloban en tres tipos. Estos son:

- Tabla en la cual el usuario escribe su nombre mediante el teclado. En este tipo de tabla el usuario suele poder introducir entre 8 y 15 letras para poner su nombre. El nombre se escribe pulsando las teclas que tienen las letras que forman el nombre.

- Tabla en la que el usuario sólo puede introducir tres caracteres mediante el teclado o palanca de juegos. En este tipo de tabla las tres letras no se suelen introducir pulsando las teclas que correspondan, sino que se suelen utilizar las teclas de control del juego para introducirlas. Por lo general, se utilizan las teclas de movimiento a la derecha, a la izquierda y la de disparo. La tecla de movimiento hacia la derecha sirve para hacer que las letras vayan avanzando. La tecla que controla el movimiento hacia la izquierda sirve para hacer que las letras vayan retrocediendo. La tecla de disparo sirve para fijar la letra en cuestión y pasar a la siguiente. Este tipo es bastante poco recomendable, porque no suele gustar al usuario y porque no permite la in-

roducción de más de tres letras. Por supuesto, se puede hacer que el programa nos deje introducir 10 ó 15 letras, pero el método empleado para su fijación es bastante incómodo para el usuario.

— El tercer tipo de tabla es mucho más imaginativo y da unos resultados muy buenos. Por una parte, en un trozo de la pantalla se imprimen los RECORDS y los nombres de los jugadores que ostentan dichos RECORDS. En otra parte de la pantalla se imprimen toda las letras que el jugador puede utilizar. La forma de escribir el nombre se puede hacer de muchas formas. Moviendo un cursor que se coloque encima de la letra a elegir y pulsando la tecla de disparo; moviendo dos cursores en cuya intersección se encuentra la letra elegida y pulsar el disparo, etc. Cualquiera de las maneras que elijamos puede ir aderezada de sonidos o de otros efectos especiales.

Una vez visto qué tipos de tabla podemos realizar para nuestros juegos y comprendido que es una de las partes más importantes de los mismos, vamos a ver cómo funcionaba el programa 1.

Este programa utiliza dos vectores para almacenar la información referente a los RECORDS. Como máximo tendremos almacenados diez RECORDS. El nombre del jugador que haya conseguido un RECORD se compondrá como máximo de 15 caracteres. Los dos vectores que utilizaremos serán:

- A\$() Para almacenar los nombres.
- S() Para almacenar las puntuaciones.

En las líneas 2500 y 2510 se realiza un bucle de retardo que en verdad no forma parte de la rutina, pero que en el programa en que ésta estaba incluida era necesario.

Lo primero que vamos a hacer va a ser dibujar un rectángulo que ocupe casi toda la pantalla dentro del cual imprimiremos toda la información. Este recuadro puede y debe ser todo lo vistoso que podamos o queramos. En este caso no lo es mucho, debido a la necesidad de hacer que el programa sea compatible con todos los ordenadores.

Línea 2520. Borrarnos la pantalla e imprimimos un mensaje que le dice al jugador cuál va a ser la información que va a ver aparecer en la pantalla.

Línea 2530. Dejamos una línea en blanco e imprimimos el lado superior del rectángulo. Lo imprimimos a base de guiones (-).

Línea 2540. Comenzamos un bucle dentro del cual vamos a imprimir los dos lados laterales del rectángulo.

Línea 2550. Colocamos el cursor en la posición que nos interesa. Cada vez colocamos el cursor una línea más abajo, tal como nos va indicando la variable índice del bucle (I).

Línea 2560. Imprimimos un signo de admiración (!) en la primera columna de cada línea y otro en la última columna.

Línea 6570. Terminamos el bucle.

Línea 6580. Imprimimos la línea inferior del rectángulo también con guiones (-).

En la línea 2590 comprobamos si la puntuación que ha conseguido el jugador (en este caso viene dada por la variable SC) es mayor o igual a la última puntuación que tenemos almacenada en la tabla. Como la tabla está ordenada de mayor a menor, si SC es mayor o igual que la última puntuación de la tabla, entonces dicha puntuación merece estar incluida en la tabla de RECORDS. En caso afirmativo, hacemos que el último elemento de la tabla sea igual a la puntuación conseguida por el usuario. También hacemos que el nombre de dicho jugador sea igual al carácter 254. Uno puede pensar que nos estamos cargando el nombre y la puntuación de una persona que estaba en la tabla, pero no es verdad. Como hay que incluir el nuevo RECORD en la tabla, sea cual sea la posición de éste, como sólo podemos tener 10 RECORDS, tendremos que quitar por fuerza siempre el último de ellos. El por qué asignamos al último elemento de A\$() el valor del carácter 254 es porque, más tarde, una vez ordenados de nuevo todos los RECORDS, vamos a buscar cuál tiene por nombre el carácter 254 para preguntar al usuario cómo se llama.

El siguiente grupo de líneas se encarga de reordenar la tabla de RECORDS basándose en la puntuación.

Línea 2600. Comenzamos un bucle desde uno hasta diez dentro del cual vamos a ordenar la tabla. Necesitaremos diez vueltas para ordenarla del todo.

Línea 2610. Realizamos un bucle desde 1 hasta el valor que tenga I en ese momento. Dentro de este segundo bucle se realizará la comparación de cuál es menor. Necesitamos dos bucles para comparar todos los RECORDS de la tabla entre sí.

Línea 2620. Miramos si el RECORD al cual apunta la variable J es menor que al que

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

apunta la variable I. En caso afirmativo hay que intercambiar las posiciones de ambos RECORDS. Para ello utilizamos una variable auxiliar (A) que nos sirve para almacenar uno de los RECORDS durante un momento, para poder realizar el intercambio. Lógicamente también intercambiamos los nombres que se encuentran almacenados en el vector A\$(). Utilizamos la variable B\$ como auxiliar.

Líneas 2630 y 2640. Terminamos los dos bucles que controlan la ordenación.

A continuación tenemos un nuevo bucle gracias al cual vamos a encontrar en qué posición se ha quedado el nuevo RECORD que acabamos de almacenar.

Línea 2650. Comenzamos el bucle desde uno hasta diez, pues diez son los RECORDS que tenemos.

Línea 2660. Miramos cuál de ellos tiene por nombre al carácter 254. Una vez que lo encontramos, almacenamos su posición en la variable PP.

Línea 2670. Terminamos el bucle.

Toda esta parte del programa se ha ejecutado sin mirar realmente si había habido un nuevo RECORD; por ello, en la línea 2680, comprobamos si PP es igual a cero (0). En caso afirmativo, significa que no hubo nuevo RECORD, por lo que vamos a la línea 2700 para saltarnos la que viene a continuación.

En la línea 2690 hacemos que el elemento de A\$() que corresponde a la posición PP sea igual a 15 puntos. Esto se hace así para que, a la hora de imprimir la tabla de RECORDS, en el lugar donde el jugador tiene que poner su nombre, aparezca una serie de puntos.

El grupo de líneas que vienen a continuación nos van a imprimir toda la tabla de RECORDS en la pantalla. Primero imprimiremos el nombre y a su lado la puntuación obtenida por esa persona.

Línea 2700. Como tenemos 10 RECORDS, realizamos un bucle de uno a diez.

Línea 2710. Colocamos el cursor en la tercera columna de la línea 5+1.

Línea 2720. Imprimimos el nombre, y más adelante la puntuación obtenida por dicha persona.

Línea 2730. Terminamos el bucle.

Las líneas que vienen a continuación permitirán que el jugador introduzca su nombre tecleándolo desde el teclado.

Línea 2740. Colocamos el cursor en la tercera columna de la línea donde está el RECORD del jugador. Esto lo podemos hacer porque conocemos su posición en la pantalla gracias a la variable PP que almacenaba la posición del nuevo RECORD en la lista.

Línea 2750. Borrarnos toda la información que había en A\$(PP), esto es necesario porque ahora el jugador va a introducir su nombre y dicho nombre tiene que estar almacenado en A\$(PP).

Línea 2760. Volvemos a preguntar si PP es igual a cero; en caso afirmativo, como ya hemos dicho, es que no ha habido nuevo RECORD. En tal caso nos vamos a la línea 2850 y nos saltamos la parte del programa que permite la introducción del nombre.

Línea 2770. Realizamos un bucle desde uno hasta quince dentro del cual iremos recogiendo, letra a letra, el nombre del jugador. El bucle se repite quince veces porque dicho nombre tiene que tener como máximo quince letras.

Línea 2780. Miramos si el jugador ha pulsado alguna tecla. En caso negativo volvemos a mirar hasta que lo haga.

Línea 2790. Si el usuario quería borrar la última letra escrita y había alguna letra escrita, la borramos de la pantalla, imprimimos en su lugar un punto, le restamos uno al contador de caracteres introducidos y quitamos de A\$(PP) la última letra introducida.

Línea 2800. Si pulsó la barra de espacio nos vamos a la línea 2830 para procesarla.

Línea 2810. Si pulsó la tecla RETURN (o ENTER) es que el jugador ha terminado de escribir su nombre. Por ello, ponemos el contador de caracteres introducidos (I) a 15, y nos vamos a la línea 2840.

Línea 2820. Si el jugador pulsó una tecla con código ASCII menor que cero (0) o mayor que zeta (z), entonces nos volvemos a la línea 2780 porque la tecla pulsada no es válida. El carácter espacio en blanco y retorno de carro tienen un código ASCII menor que el cero, por ello se pregunta por ellos antes de llegar a esta línea, para que puedan ser válidos.

Línea 2830. Imprimimos la tecla pulsada, dejando el cursor al lado de ella y concatenamos A\$(PP) con la tecla pulsada.

Línea 2840. Terminamos el bucle y nos vamos a recoger la siguiente tecla del teclado, siempre que no hayamos tecleado las quince permitidas, en cuyo caso el bucle se acaba y pasamos a la siguiente línea.

Línea 2850. Hacemos desaparecer el cursor que antes habíamos hecho visible.

Línea 2860. Colocamos el cursor en la línea 20.

Línea 2870. Imprimimos un mensaje para que el jugador pulse una tecla cuando esté listo.

Línea 2880. Esperamos hasta que se pulse una tecla cualquiera.

A partir de esta línea podemos considerar que la rutina se ha terminado. Aunque quedan dos líneas más, éstas pertenecen a la es-

tructura del programa dentro del cual está incluida esta rutina.

Hemos hablado de los tres tipos de rutinas que podemos realizar para crear nuestras propias tablas de RECORDS en nuestros programas. A continuación vamos a ver un ejemplo del segundo tipo. En este caso el usuario sólo podrá introducir tres letras utilizando las teclas de movimiento. Estas teclas serán:

O Izquierda
P Derecha
M Fuego

```
2450 REM
2455 REM *****
2460 REM * TABLA DE GANADORES *
2465 REM *****
2470 REM
2475 FOR I=1 TO 1000
2480 NEXT I
2485 CLS
2490 PRINT "PUNTUACIONES DE HOY"
2495 PRINT
2500 PRINT "-----"
2505 FOR I=4 TO 17
2510     PRINT " ";TAB(40);"! "
2515 NEXT I
2520 PRINT "-----"
2525 IF SC=S(10) THEN LET S(10)=SC:LET A$(10)=CHR$(254)
2530 FOR I=1 TO 10
2535     FOR J=1 TO I
2540         IF S(J)<S(I) THEN LET A=S(J):LET S(J)=S(I):LET S(I)=A:LET B$=A$(J):LE
T A$(J)=A$(I):LET A$(I)=B$
2545     NEXT J
2550 NEXT I
2555 FOR I=1 TO 10
2560     IF A$(I)=CHR$(254) THEN LET PP=I
2565 NEXT I
2570 IF PP=0 THEN GOTO 2715
2575 LET A$(PP)="***"
2580 FOR I=1 TO 10
2585     LOCATE 5+I,3
2590     PRINT A$(I);".....";TAB(30);S(I)
2595 NEXT I
2600 IF PP=0 THEN GOTO 2715
2605 LOCATE 5+PP,3
2610 LET A$(PP)=" "
2615 LET C=2
2620 LET X=0
2625 LET Z$=" ABCDEFGHIJKLMNOPQRSTUVWXYZ.,;:!?#$/&*+-=<>"
2630 FOR I=1 TO 3
2635     LOCATE 5+PP,2+I
2640     PRINT MID$(Z$,C,1)
2645     LET B$=INKEY$:IF B$="" THEN GOTO 2645
2650     IF B$="P" OR B$="M" THEN LET C=C+1:GOTO 2670
```

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```

2655 IF B$="O" OR B$="o" THEN LET C=C-1:GOTO 2670
2660 IF B$="M" OR B$="m" THEN LET X=-1:GOTO 2670
2665 GOTO 2645
2670 LOCATE 5+PP,2+I
2675 IF X=-1 THEN GOTO 2700
2680 IF C>LEN(Z$) THEN LET C=1
2685 IF C<1 THEN LET C=LEN(Z$)
2690 PRINT MID$(Z$,C,1)
2695 GOTO 2645
2700 LET A$(PP)=A$(PP)+MID$(Z$,C,1)
2705 LET X=0
2710 NEXT I
2715 LOCATE 20,1
2720 PRINT "<< PULSA UNA TECLA >>"
2725 LET B$=INKEY$
2730 IF B$="" THEN GOTO 2725
2735 LOCATE 20,1
2740 GOTO 2330

```

Si ejecutamos esta rutina tal y como está, no vamos a conseguir mucho, ya que no hemos dimensionado los vectores que utilizamos y tampoco hemos introducido los nombres y las puntuaciones ficticias de las que hablábamos. Si juntamos el programa número dos con el número tres y hacemos RUN, veremos cómo funciona. Otra solución es introdu-

cir esta rutina en el programa del GUSANO LOCO en las mismas líneas que ocupaba la rutina que acabamos de explicar. Aunque ambas rutinas comparten bastantes números de línea, si incluimos esta rutina en el programa GUSANO LOCO tendremos que borrar las líneas que van desde la 2750 hasta la 2900 en el programa original.

```

100 REM *****
110 REM * PROGRAMA DEMO PARA VER *
120 REM * EL FUNCIONAMIENTO DE *
130 REM * LA RUTINA DE RECORDS. *
140 REM *****
150 REM
160 CLS
170 DIM A$(10)
180 DIM S(10)
190 LET PP=0
200 FOR I=1 TO 10
210   READ A$(I),S(I)
220 NEXT I
230 INPUT "QUE PUNTUACION QUIERES TENER ";SC
240 GOSUB 2450
250 CLS
260 PRINT "FIN DE LA DEMOSTRACION"
270 END
280 REM
290 REM *****
300 REM * DATAS CON NOMBRES Y PUNTUACIONES FICTICIAS *
310 REM *****
320 REM
330 DATA "FMG",12455
340 DATA "JYH",11022

```

```

350 DATA "ABA",10344
360 DATA "RIP",9800
370 DATA "PUS",9780
380 DATA "RES",9201
390 DATA " A ",7200
400 DATA "Z Z",6543
410 DATA "ZIP",3403
420 DATA "ZAP",2195

```

Las modificaciones que hay que realizar en el programa número 2 para que pueda funcionar en ordenadores distintos del IBM son las siguientes:

COMMODORE:

```

2485 PRINT CHR$(147)
2510 PRINT "!";TAB(39);"!"
2585 POKE 214,5+I:POKE 211,3
2605 POKE 214,5+PP:POKE 211,3
2635 POKE 214,5+PP:POKE 211,2+I
2645 GET B$:IF B$="" THEN GOTO 2645
2670 POKE 214,5+PP:POKE 211,2+I
2715 POKE 214,20:POKE 211,1
2725 GET B$
2735 POKE 214,20:POKE 211,1

```

SPECTRUM:

```

2510 PRINT "!";TAB(31);"!"
2585 PRINT AT 5+I,3;
2590 PRINT A$(I);".....";TAB(25);S(I)
2605 PRINT AT 5+PP,3;
2635 PRINT AT 5+PP,2+I;
2640 PRINT Z$(C)

```

```

2670 PRINT AT 5+PP,2+I;
2690 PRINT Z$(C)
2700 LET A$(PP)=A$(PP)+Z$(C)
2715 PRINT AT 20,0;
2735 PRINT AT 20,0;

```

MSX Y AMSTRAD:

```

2585 LOCATE 3,5+I
2605 LOCATE 3,5+PP
2635 LOCATE 2+I,5+PP
2670 LOCATE 2+I,5+PP
2715 LOCATE 1,20
2735 LOCATE 1,20

```

A continuación aparecen las modificaciones que hay que realizar en el tercer programa para que funcione en el COMMODORE y en SPECTRUM.

COMMODORE:

```

160 PRINT CHR$(147)
250 PRINT CHR$(147)

```

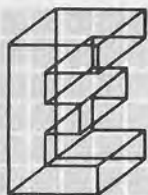
SPECTRUM:

```

170 DIM A$(10,3)

```

TRUCOS Y RUTINAS BASICAS



N el tomo anterior vimos una serie de programas que nos permitían la realización de letras más grandes de las normales con nuestro ordenador. Se propusieron una serie de cambios en estos programas, pero no se ex-

plicó nada acerca de ellos. A continuación vamos a explicar cuál era su funcionamiento y vamos a ver algunos programas más que realizan la misma función, pero de diferente forma.

Primero vamos a explicar el primer programa. Este nos permitía imprimir mensajes ocho veces más grandes de lo normal, pero era excesivamente lento. Veamos su funcionamiento.

Para que el programa funcionase teníamos que almacenar en A\$ el mensaje que queríamos imprimir. También había que darle las coordenadas de la pantalla donde queríamos que se imprimiese el mensaje. Esto lo hacíamos dándole valores a X y a Y.

Línea 6230. Comenzamos un bucle que va desde uno (1) hasta la longitud del mensaje a imprimir. Dentro de este bucle vamos a imprimir dicho mensaje letra a letra.

Línea 6240. Hacíamos un RESTORE a las líneas de DATAS para colocar el puntero de lectura al principio de ellas.

Línea 6250. Almacenamos en la variable numérica N el código ASCII de la letra que tenemos que imprimir.

Línea 6260. Le restamos 32 a N porque no tenemos definidos los primeros 31 caracteres de la tabla. Hay que recordar que estos caracteres son de control y no son visibles como caracteres propiamente dichos.

Línea 6270. Como cada definición de cada carácter se compone de ocho bytes, tenemos que multiplicar el número almacenado en N por ocho para saber en qué posición de las líneas DATA se encuentra su definición.

Líneas 6280 a 6300. Como no podemos colocarnos en la línea de DATA que nos interese de una forma automática, tenemos que leer todos los valores que se encuentran antes de los que definen el carácter que tenemos que imprimir, para colocar el puntero en el lugar que nosotros deseamos. Estas son las líneas que van a hacer que el programa sea demasiado lento. Por supuesto, podríamos colocar el puntero de las líneas DATA en el lugar apropiado usando la instrucción RESTORE, pero como tenemos muchas letras definidas, necesitaríamos muchas líneas RESTORE para hacerlo.

Línea 6310. Una vez que tenemos el puntero colocado en la posición de las líneas DATA donde empieza la definición del carácter que queremos imprimir, comenzamos un bucle de uno a ocho dentro del cual vamos a leer cada uno de estos valores, vamos a pasarlo a binario y vamos a dibujar cada línea del carácter.

Línea 6320. Leemos cada byte de la definición.

Línea 6330. Colocamos el cursor donde corresponde.

Líneas 6340 a 6490. En este grupo tan largo de líneas vamos comprobando que un determinado bit del byte que estamos utilizando está encendido o apagado. Si está encendido, imprimiremos un cuadrado en la pantalla y si éste está apagado, imprimiremos un blanco. Para ello miramos si el valor del byte es igual o superior a un cierto valor. Cada va-

lor corresponde al valor de un bit en el byte. Cada vez que nos encontramos con que uno de esos valores existe, imprimimos un cuadrado y restamos al byte el valor de dicho bit. Si dicho valor no existe, entonces, como dicho bit está a cero (0), imprimimos un espacio en blanco.

Línea 6500. Como se utilizó el punto y coma para que el ordenador no cambiase de línea hasta que se hubiese impreso una línea entera, hacemos que cambie de línea con un PRINT sin punto y coma (;).

Línea 6510. Terminamos el bucle.

Línea 6520. Cuando el ordenador llega a esta línea de programa, ya ha impreso una letra entera. Como la siguiente letra tiene que estar a la derecha de la anterior y como cada letra ocupa ocho caracteres, tenemos que sumar ocho a la variable X que nos dice en qué columna tenemos que imprimir.

Línea 6530. Si hemos llegado al borde de la pantalla hacemos que X tenga valor uno (1) y sumamos ocho a la variable Y para que la continuación del mensaje se imprima ocho líneas más abajo.

Línea 6540. Aquí termina el segundo bucle para continuar con la siguiente letra del mensaje a imprimir.

Línea 6550. Devolvemos el control al programa principal.

Líneas 8000 a 9020. Se encuentran las líneas DATA que almacenan las definiciones de las letras que van desde el espacio en blanco hasta el último carácter que no es semigráfico. Estas letras son:

```
!"#$%&'()*+,-./0123456789;<=>? ABCDEFGH-  
IJKLMNOPQRSTUVWXYZ[\]^_`'abcdefghijklmnopqrstuvwxyz{|
```

A continuación vimos un programa que nos permitía visualizar todos los caracteres que habíamos definido para ver si alguno de ellos no estaba bien y nos habíamos equivocado al introducir las líneas DATA.

El tercero de los programas era una demostración para que el usuario introdujese sus propios mensajes y viese cómo aparecían en pantalla.

De estos dos últimos programas no hace falta explicar nada debido a su simplicidad.

El programa número cuatro nos permitiría imprimir los caracteres invertidos. Esto es, dibujar su contorno en vez de su figura. Esto lo conseguíamos imprimiendo un cubo cuando un bit fuese cero e imprimiendo un es-

pacio en blanco cuando dicho bit fuera un uno. Como se puede apreciar, esta operación es la inversa de la que realiza el primer programa.

El siguiente programa (el número 5) imprimía los caracteres con asteriscos (*) en vez de con cuadrados. La forma era la misma que en el programa 1. Si un bit estaba a uno, imprimíamos un asterisco y si estaba a cero, un espacio en blanco.

El sexto hacía que cada letra estuviese formada por sí misma. Esto es, si imprimíamos una 'A', ésta estaba formada por A's. Si imprimíamos una 'j', ésta estaba formada por j's. La forma de hacer esto era definir una variable alfanumérica B\$ al principio de cada bucle y cuyo valor tenía que ser la letra que queríamos representar. A la hora de dibujar la letra, en vez de utilizar cuadraditos utilizábamos el valor de B\$.

Por fin, y ya para terminar con los programas vistos en el tomo anterior, realizamos otro programa de impresión de caracteres gigantes que era mucho más rápido. La única dificultad que tenía era que ocupaba demasiada memoria.

Su funcionamiento es igual que el programa 1. En A\$ tenemos que almacenar el mensaje a imprimir y en X y en Y las coordenadas donde queremos que empiece el mensaje. Este programa almacenaba todas las definiciones de los caracteres en una matriz llamada D(). Con ello ganábamos velocidad al dibujar un carácter, y que el acceso a una variable es inmediato y mucho más rápido que a las líneas DATA. Lo único que había que hacer para utilizar esta rutina era llamar a la línea 6000 antes de ejecutarla por primera vez para que el ordenador cargase en el vector D() la definición de todos los caracteres. Una vez hecho esto, si queríamos utilizar la rutina sólo teníamos que llamar a la línea 7000.

El funcionamiento de este programa es muy parecido al primero. La única diferencia está en que al principio leemos todos los valores de las líneas DATA y que en vez de buscar el carácter a definir en estas líneas lo buscamos en una matriz. La matriz está compuesta de 96 x 8 elementos. El primer número nos da el número de carácter menos 31. La segunda dimensión almacena cada una de las ocho filas que definen dicho carácter.

A continuación vamos a ver un programa que siendo igual de rápido que el último que acabamos de ver, utiliza menos memo-

TRUCOS Y RUTINAS BASICAS

ria para el almacenamiento de los caracteres. La forma de encontrar la posición de un carácter es más complicada, pero no se pierde

velocidad y, como se gana memoria, hace que nuestra rutina sea más práctica.

```

6000 REM *****
6010 REM *
6020 REM *      <<< IMPRESION DE CARACTERES GIGANTES >>>
6030 REM *
6040 REM * VARIABLES QUE HAY QUE PASARLE A LA RUTINA
6050 REM * -----
6060 REM *
6070 REM * A$ = MENSAJE A IMPRIMIR
6080 REM * X = COLUMNA DONDE EMPEZARA EL MENSAJE
6090 REM * Y = FILA DONDE SE COLOCARA EL MENSAJE
6100 REM *
6110 REM * VARIABLES USADAS POR LA RUTINA
6120 REM * -----
6130 REM *
6140 REM * T = CONTADOR PARA SABER QUE LAS DATAS SON CORRECTAS*
6150 REM * A = VARIABLE QUE RECOGE EL DATO DE LA MATRIZ
6160 REM * I = VARIABLE INDICE DEL PRIMER BUCLE
6170 REM * J = VARIABLE INDICE DEL SEGUNDO BUCLE
6180 REM * N()= MATRIZ QUE ALMACENA LAS DEFINICIONES
6190 REM * L = LINEA EN LA QUE ESTA LA DEFINICION
6200 REM * P = POSICION DE LA DEFINICION EN DICHA LINEA
6210 REM * N = CODIGO ASCII DE LA LETRA A IMPRIMIR
6220 REM * B$ = BITE A REPRESENTAR EN HEXADECIMAL
6230 REM *
6240 REM * LA RUTINA NO DEVUELVE NINGUN VALOR
6250 REM *
6260 REM *****
6270 REM
6280 REM *****
6290 REM * LECTURA DE LAS DATAS Y ALMACENAMIENTO *
6300 REM *****
6310 REM
6320 DEF FNZ=16*(INSTR(Z$,MID$(B$,1,1))-1)+INSTR(Z$,MID$(B$,2,1))-1
6330 LET Z$="0123456789ABCDEF"
6340 DIM N$(24)
6350 LET T=0
6360 LET L=3070
6370 FOR I=1 TO 24
6380   READ A$,A
6390   FOR J=1 TO LEN(A$) STEP 2
6400     LET B$=MID$(A$,J,2)
6410     LET T=T+FNZ
6420   NEXT J
6430   IF T<>A THEN PRINT "ERROR EN LA LINEA ";L:END
6440   LET T=0
6450   LET L=L+20
6460   LET N$(I)=A$
6470 NEXT I
6480 RETURN
7000 REM
7010 REM *****
7020 REM * PROGRAMA PRINCIPAL *
7030 REM *****
7040 REM
7050 FOR I=1 TO LEN(A$)
7060   LET N=ASC(MID$(A$,I,1))
7070   LET N=N-31
7080   LET L=INT(N/4)+1
7090   LET P=N-4*(L-1)

```

```

7100 LET P=(P 1)*16+1
7110 IF P<0 THEN LET L=L-1:LET P=49
7120 FOR J=1 TO 8
7130 LOCATE Y+J-1,X
7140 LET B$=MID$(N$(L),P+2*J-2,2)
7150 LET A=FNZ
7160 IF A>=128 THEN LET A=A-128:PRINT CHR$(219);:GOTO 7180
7170 PRINT " ";
7180 IF A>=64 THEN LET A=A-64:PRINT CHR$(219);:GOTO 7200
7190 PRINT " ";
7200 IF A>=32 THEN LET A=A-32:PRINT CHR$(219);:GOTO 7220
7210 PRINT " ";
7220 IF A>=16 THEN LET A=A-16:PRINT CHR$(219);:GOTO 7240
7230 PRINT " ";
7240 IF A>=8 THEN LET A=A-8:PRINT CHR$(219);:GOTO 7260
7250 PRINT " ";
7260 IF A>=4 THEN LET A=A-4:PRINT CHR$(219);:GOTO 7280
7270 PRINT " ";
7280 IF A>=2 THEN LET A=A-2:PRINT CHR$(219);:GOTO 7300
7290 PRINT " ";
7300 IF A=1 THEN PRINT CHR$(219);:GOTO 7320
7310 PRINT " ";
7320 PRINT
7330 NEXT J
7340 LET X=X+8
7350 IF X>80 THEN LET X=1:LET Y=Y+8
7360 NEXT I
8000 REM
8010 REM *****
8020 REM * LINEAS DE DATAS CON NUMEROS *
8030 REM * EN HEXADECIMAL CON LA DEF1- *
8040 REM * NICION DE LOS CARACTERES. *
8050 REM *****
8060 REM
8070 DATA "000000000000000018181818180018006C6C6C00000000006C6CFE7CFE6C6C00"
8080 DATA 1532
8090 DATA "183E583C1A7C18000063660C18336300386C3876DCDC76001818300000000000"
8100 DATA 1787
8110 DATA "0C18303030180C0030180C0C0C18300000663CFF3C6600000018187E18180000"
8120 DATA 1197
8130 DATA "00000000001818300000007E000000000000000000181800060C183060C08000"
8140 DATA 776
8150 DATA "7CC6CED6E6C67C001838181818187E003C66063C60667E003C46061C06663C00"
8160 DATA 2480
8170 DATA "18385898FE183C007E62603C06663C003C66607C66663C007E46060C18181800"
8180 DATA 2138
8190 DATA "3C66663C66663C003C66663E06663C0000001818001818000000181800181830"
8200 DATA 1322
8210 DATA "0C18306030180C0000007E000007E00006030180C183060003C66060C18001800"
8220 DATA 1092
8230 DATA "7CC6DEDEDEC07C00183C66667E666600FC66667C6666FC003C66C0C0C06663C00"
8240 DATA 3858
8250 DATA "F86C6666666CF800FE6268786862FE00FE6268786860F0003C66C0C0CEC67E00"
8260 DATA 4142
8270 DATA "6666667E666666007E18181818187E001E0C0C0CCCC7800E6666C786C66E600"
8280 DATA 2704
8290 DATA "F06060606266FE00C6EEFED6D6C6C600C6E6F6DECEC6C600386CC6C6C66C3800"
8300 DATA 4916
8310 DATA "FC6666786060F000386CC6C6DACC7600FC66667C6666E2003C66603C06663C00"
8320 DATA 3610
8330 DATA "7E5A181818183C006666666666663C0066666666663C1800C6C6C6D6FEEEC600"
8340 DATA 3136
8350 DATA "C66C38386CC6C60066666663C18183C00FEC68C183266FE003C303030303030C00"
8360 DATA 2778
8370 DATA "C06030180C0603013C0C0C0C0C0C0C00183C7E181818180000000000000000FF"
8380 DATA 1123

```

TRUCOS Y RUTINAS BASICAS

```

8390 DATA "0030180C000000000000C0780C7CCC7600E0E07C666666BC0000003C6660663C00"
8400 DATA 2148
8410 DATA "1C0C7CCCCCCC760000003C667E603C001C3630783030780000003E66663E067C"
8420 DATA 2282
8430 DATA "E0606C766666E6001800381818183C0002000E060666663CE060666C786CE600"
8440 DATA 2472
8450 DATA "3818181818183C0000006CFED6D6C6000000D866666660000003C6666663C00"
8460 DATA 2274
8470 DATA "0000DC66667C80F0000076CCCC7C0C1E0000D86C8060F00000003C603C067C00"
8480 DATA 2678
8490 DATA "30307C3030361C000000666666663E0000006666663C18000000C6D6D6FE6C00"
8500 DATA 2246
8510 DATA "0000C66C386CC60000006666663E067C00007E4C18307E000E18187018180E00"
8520 DATA 1802
8530 DATA "18181818181818007018180E1818700076D8000000000000CC33CC33CC33CC33"
8540 DATA 1856

```

Como se puede apreciar, la definición de los caracteres está en hexadecimal. Con ello ganamos bastante, ya que un número en hexadecimal siempre ocupa dos bytes en las líneas DATA y los números en decimal ocupan por lo menos dos y puede llegar a ocupar cuatro. Si uno se fija en las líneas DATA, verá que éstas son una ristra sin sentido de números y letras sin comas. Cada línea DATA ocupa 64 bytes y almacena 32 números en notación hexadecimal. De momento ya hemos ahorrado algo de memoria.

Por otra parte, a la hora de almacenar estas ristras de números y letras en la memoria, lo haremos dentro de un vector alfanumérico. Los vectores alfanuméricos ocupan, por lo general, menos memoria que las tablas numéricas. Realizando este tipo de almacenamiento también ahorramos memoria.

El programa funciona en el IBM y compatibles; para los demás ordenadores las modificaciones que hay que realizar son:

COMMODORE:

```

6320 REM
6410 GOSUB 7500:LET T=T+R
7130 POKE 214,Y+J-1:POKE 211,X
7150 GOSUB 7500:LET A=R
7350 IF X>40 THEN LET X=1:LET Y=Y-8

```

Hay que sustituir todos los CHR\$(219) por CHR\$(166), tal y como se hizo en el tomo anterior.

También hay que incluir el siguiente grupo de líneas:

```

7500 REM
7510 REM *****
7520 REM * PASO DE HEXADECIMAL A DECIMAL *
7530 REM *****
7540 REM

```

```

7550 LET R=0
7560 FOR K=1 TO 2
7570 FOR O=1 TO 16
7580 IF MID$(Z$,O,1)<>MID$(B$,K,1) THEN GOTO 7600
7590 LET R=R+INT((O-1)*(16^(2-K))+.5:LET O=16
7600 NEXT O
7610 NEXT K
7620 RETURN

```

SPECTRUM:

```

6320 REM
6340 DIM N$(24,64)
6400 LET B$=A$(J TO J+1)
6410 GOSUB 7500:LET T=T+R
7060 LET N=CODE(A$(I))
7130 PRINT AT Y+J-1,X;
7140 LET B$=N$(L,P+2*J-2 TO P+2*J-1)
7150 GOSUB 7500:LET A=R
7350 IF X>32 THEN LET X=1:LET Y=Y+8

```

También hay que cambiar todos los CHR\$(219) por CHR\$(143), como se vio en el tomo anterior.

Hay que unir el programa con las siguientes líneas:

```

7500 REM
7510 REM *****
7520 REM * PASO DE HEXADECIMAL A DECIMAL *
7530 REM *****
7540 REM
7550 LET R=0
7560 FOR K=1 TO 2
7570 FOR O=1 TO 16
7580 IF Z$(O)<>B$(K) THEN GOTO 7600
7590 LET R=R+INT((O-1)*(16^(2-K))+.5:LET O=16
7600 NEXT O
7610 NEXT K
7620 RETURN

```

MSX:

```

7130 LOCATE X,Y+J-1
7350 IF X>40 THEN LET X=1:LET Y=Y+8

```

AMSTRAD:

```

7130 LOCATE X,Y+J-1

```

Si no se utiliza el programa en el modo 2 de pantalla, hay que variar la siguiente línea:

```
7530 IF X>40 THEN X=1:Y=Y+8:REM PARA EL MODO 1
7350 IF X>32 THEN X=1:Y=Y+8:REM PARA EL MODO 0
```

El funcionamiento del programa línea a línea es el siguiente:

Entre las líneas 6000 y 6480 se almacena el contenido de las líneas DATA en el vector alfanumérico N\$(). También se realiza una comprobación numérica para ver si el usuario ha introducido correctamente las líneas DATAS. En el caso de que hubiese algún error en ellas, el programa lo indicaría y nos diría en qué línea.

Línea 6320. Se define una función que se encargará de pasarnos los números en hexadecimal a decimal. En el COMMODORE y en el SPECTRUM, como no tienen la función INSTR, se ha realizado una subrutina aparte, que empieza a partir de la línea 7500 para que nos haga dicho cambio. La explicación de dicha rutina puede encontrarse en el segundo tomo de esta colección.

Línea 6330. Definimos una variable alfanumérica (Z\$) que almacena todos los dígitos posibles en hexadecimal. Esta variable será utilizada por la función de la línea de arriba.

Línea 6340. Definimos el vector que almacenará toda la información de las líneas DATA.

Línea 6350. Ponemos la variable T a cero. Esta variable será la encargada de sumar los números que hay almacenados en hexadecimal en las líneas DATA. Después de almacenar cada línea, compararemos el valor de T con el número de control que también están en las líneas DATA para saber si son iguales. En el caso de que sean distintos, imprimiremos un mensaje de error.

Línea 6360. Almacenamos en la variable L el valor 8070. Este número es el número de línea donde se encuentra la primera línea DATA. Gracias a esta variable podremos decirle al usuario qué línea tiene una DATA incorrecta.

Línea 6370. Comenzamos un bucle desde 1 hasta 24 dentro del cual leeremos las líneas DATA. Como hay 24 líneas DATA, tenemos que leer 24 veces.

Línea 6380. Leemos la cadena en hexadecimal y el número de control.

Línea 6390. Comenzamos un bucle dentro del cual vamos a transformar todos los números en hexadecimal que hemos leído en decimal y vamos a ver si su suma coincide con el número de control.

Línea 6400. Cogemos los caracteres de la cadena A\$ de dos en dos y los almacenamos en B\$.

Línea 6410. Sumamos a T el valor de dichos números en decimal.

Línea 6420. Y terminamos el bucle.

Línea 6430. Si la suma es distinta del control, imprimimos un mensaje de error y decimos en qué línea se encuentra. También terminamos el programa.

Línea 6440. Si el número coincide con el control, hacemos que uno sea igual a cero, para continuar con la siguiente línea de DATA.

Línea 6450. Sumamos 20 al contador de líneas. Esto se hace así porque las líneas DATA que contienen la definición de los caracteres en hexadecimal se encuentran en números de línea que tienen una diferencia de 20 entre una y otra.

Línea 6460. Almacenamos la línea de DATA leída en el vector.

Línea 6470. Terminamos el bucle para leer la siguiente línea.

Línea 6480. Y devolvemos el control al programa principal.

Esta parte del programa tiene que ser llamada antes de ejecutar la rutina de impresión de letras por primera vez. Tarda un poco de tiempo en ejecutarse debido a la comprobación que hace de los números introducidos. Para que no tarde tanto, una vez introducido el programa y ejecutada esta rutina, si no ha dado errores en ninguna línea DATA, se pueden suprimir las líneas comprendidas entre la 6390 y 6470, ambas inclusive y a excepción de la línea 6460, para ganar velocidad.

El grupo de línea que aparece a continuación son las que se encargan de dibujar el mensaje. Dentro de este grupo de líneas hay unas cuantas que ya nos suenan por haberlas utilizado en los demás programas. Por ello, sólo explicaremos las diferencias. Las líneas diferentes son las que se encargan de calcular el elemento del vector donde se encuentra la definición de una letra y su posición dentro de dicho elemento.

Línea 7050. Comenzamos un bucle para ir analizando y dibujando letra a letra el mensaje almacenado en A\$.

TRUCOS Y RUTINAS BASICAS

Línea 7060. Almacenamos en N el código ASCII de la letra a dibujar en la vuelta del bucle en la que nos encontramos.

Línea 7070. Como siempre, restamos 31 del valor de N. Hacemos esto porque los primeros 31 caracteres del código ASCII no son imprimibles.

Línea 7080. Calculamos el elemento del vector N\$() que contiene la definición de dicho carácter. Dividimos el valor de N entre cuatro porque en cada elemento del vector están definidos cuatro caracteres.

Línea 7090. Hacemos que P tenga el resto existente entre N y el número del elemento del vector.

Línea 7100. Almacenamos en P la posición dentro del elemento L del vector donde se encuentra la definición del carácter a dibujar.

Línea 7110. Cuando el carácter a dibujar es el último de uno de los elementos, el valor de P es menor que cero. En ese caso hacemos que P apunte a la posición donde empieza la definición del último de los caracte-

res que hay en un elemento y le restamos uno al número de elementos que contiene la definición.

A partir de este momento el programa es igual a lo que hemos visto hasta ahora.

El siguiente programa que vamos a ver nos permitirá imprimir textos de diferentes formas dependiendo del valor que le demos a la variable SW. Los valores aceptados están comprendidos entre el 1 y el 6 y cada uno de ellos nos imprime el mensaje de una manera distinta. El significado de cada número es el siguiente:

1. Impresión normal.
2. Impresión invertida.
3. Impresión normal con un carácter distinto del cuadradito.
4. Impresión invertida con un carácter distinto del cuadradito.
5. Impresión normal. Se imprime cada carácter con el carácter que representa.
6. Impresión invertida. Se imprime cada carácter con el carácter que representa.

```

6000 REM *****
6010 REM *
6020 REM * <<< IMPRESION DE CARACTERES GIGANTES >>> *
6030 REM *
6040 REM * VARIABLES QUE HAY QUE PASARLE A LA RUTINA
6050 REM * -----
6060 REM *
6070 REM * A$ = MENSAJE A IMPRIMIR
6080 REM * X = COLUMNA DONDE EMPEZARA EL MENSAJE
6090 REM * Y = FILA DONDE SE COLOCARA EL MENSAJE
6100 REM * SW = INDICADOR DEL TIPO DE IMPRESION
6110 REM *
6120 REM * VARIABLES USADAS POR LA RUTINA
6130 REM * -----
6140 REM *
6150 REM * N = POSICION DE LA DEFINICION DEL CARACTER
6160 REM * A = VARIABLE QUE RECOGE EL DATO DE LA MATRIZ
6170 REM * I = VARIABLE INDICE DEL PRIMER BUCLE
6180 REM * J = VARIABLE INDICE DEL SEGUNDO BUCLE
6190 REM * N( ) = MATRIZ QUE ALMACENA LAS DEFINICIONES
6200 REM * M$ = ALMACENA EL CARACTER DE LA FORMA DE LA LETRA
6210 REM * W$ = ALMACENA EL CARACTER DEL FONDO DE LA LETRA
6220 REM *
6230 REM * LA RUTINA NO DEVUELVE NINGUN VALOR
6240 REM *
6250 REM *****
6260 REM
6270 REM *** LECTURA DE LAS DATAS ***
6280 REM
6290 DIM D(96,8)
6300 FOR I=1 TO 96
6310   FOR J=1 TO 8
6320     READ D(I,J)
6330   NEXT J

```

```

6340 NEXT I
6350 RETURN
7000 REM
7010 REM *** RUTINA PRINCIPAL ***
7020 REM
7030 IF SW<2 THEN LET M$=CHR$(219):LET W$=" "
7040 IF SW=2 THEN LET M$=" ":LET W$=CHR$(219)
7050 IF SW=3 THEN W$=" "
7060 IF SW=4 THEN LET W$=M$:LET M$=" "
7070 FOR I=1 TO LEN(A$)
7080     LET N=ASC(MID$(A$,I,1))
7090     LET N=N-31
7100     IF SW=5 THEN LET M$=MID$(A$,I,1):LET W$=" "
7110     IF SW=6 THEN LET W$=MID$(A$,I,1):LET M$=" "
7120     FOR J=1 TO 8
7130         LET A=D(N,J)
7140         LOCATE Y+J-1,X
7150         IF A>=128 THEN LET A=A-128:PRINT M$;:GOTO 7170
7160         PRINT W$;
7170         IF A>=64 THEN LET A=A-64:PRINT M$;:GOTO 7190
7180         PRINT W$;
7190         IF A>=32 THEN LET A=A-32:PRINT M$;:GOTO 7210
7200         PRINT W$;
7210         IF A>=16 THEN LET A=A-16:PRINT M$;:GOTO 7230
7220         PRINT W$;
7230         IF A>=8 THEN LET A=A-8:PRINT M$;:GOTO 7250
7240         PRINT W$;
7250         IF A>=4 THEN LET A=A-4:PRINT M$;:GOTO 7270
7260         PRINT W$;
7270         IF A>=2 THEN LET A=A-2:PRINT M$;:GOTO 7290
7280         PRINT W$;
7290         IF A=1 THEN PRINT M$;:GOTO 7310
7300         PRINT W$;
7310         PRINT
7320     NEXT J
7330     LET X=X+8
7340     IF X>80 THEN LET X=1:LET Y=Y+8
7350 NEXT I
7360 RETURN
8000 REM
8010 REM *****
8020 REM * LINEAS DE DATA CON LA *
8030 REM * DEFINICION DE TODOS LOS *
8040 REM * CARACTERES. *
8050 REM *****
8060 REM
8070 DATA 0,0,0,0,0,0,0,0:REM ESPACIO
8080 DATA 24,24,24,24,24,0,24,0:REM !
8090 DATA 108,108,108,0,0,0,0,0:REM "
8100 DATA 108,108,254,124,254,108,108,0:REM #
8110 DATA 24,62,88,60,28,124,24,0:REM $
8120 DATA 0,99,102,12,24,51,99,0:REM %
8130 DATA 56,108,56,118,220,220,118,0:REM &
8140 DATA 24,24,48,0,0,0,0,0:REM '
8150 DATA 12,24,48,48,48,24,12,0:REM (
8160 DATA 48,24,12,12,12,24,48,0:REM )
8170 DATA 0,102,60,255,60,102,0,0:REM *
8180 DATA 0,24,24,126,24,24,0,0:REM +
8190 DATA 0,0,0,0,0,24,24,48:REM ,
8200 DATA 0,0,0,126,0,0,0,0:REM -
8210 DATA 0,0,0,0,0,24,24,0:REM .
8220 DATA 6,12,24,48,96,192,128,0:REM /
8230 DATA 124,198,206,214,230,198,124,0:REM 0
8240 DATA 24,56,24,24,24,24,126,0:REM 1
8250 DATA 60,102,6,60,96,102,126,0:REM 2
8260 DATA 60,70,6,28,6,102,60,0:REM 3

```

TRUCOS Y RUTINAS BASICAS

8270 DATA	24, 56, 88, 152, 254, 24, 60, 0	: REM 4
8280 DATA	126, 98, 96, 60, 6, 102, 60, 0	: REM 5
8290 DATA	60, 102, 96, 124, 102, 102, 60, 0	: REM 6
8300 DATA	126, 70, 6, 12, 24, 24, 24, 0	: REM 7
8310 DATA	60, 102, 102, 60, 102, 102, 60, 0	: REM 8
8320 DATA	60, 102, 102, 62, 6, 102, 60, 0	: REM 9
8330 DATA	0, 0, 24, 24, 0, 24, 24, 0	: REM :
8340 DATA	0, 0, 24, 24, 0, 24, 24, 48	: REM ;
8350 DATA	12, 24, 48, 96, 48, 24, 12, 0	: REM <
8360 DATA	0, 0, 126, 0, 0, 126, 0, 0	: REM =
8370 DATA	96, 48, 24, 12, 24, 48, 96, 0	: REM >
8380 DATA	60, 102, 6, 12, 24, 0, 24, 0	: REM ?
8390 DATA	124, 198, 222, 222, 222, 192, 124, 0	: REM @
8400 DATA	24, 60, 102, 102, 126, 102, 102, 0	: REM A
8410 DATA	252, 102, 102, 124, 102, 102, 252, 0	: REM B
8420 DATA	60, 102, 192, 192, 192, 102, 60, 0	: REM C
8430 DATA	248, 108, 102, 102, 102, 108, 248, 0	: REM D
8440 DATA	254, 98, 104, 120, 104, 98, 254, 0	: REM E
8450 DATA	254, 98, 104, 120, 104, 96, 240, 0	: REM F
8460 DATA	60, 102, 192, 192, 206, 198, 126, 0	: REM G
8470 DATA	102, 102, 102, 126, 102, 102, 102, 0	: REM H
8480 DATA	126, 24, 24, 24, 24, 24, 126, 0	: REM I
8490 DATA	30, 12, 12, 12, 204, 204, 120, 0	: REM J
8500 DATA	230, 102, 108, 120, 108, 102, 230, 0	: REM K
8510 DATA	240, 96, 96, 96, 98, 102, 254, 0	: REM L
8520 DATA	198, 238, 254, 214, 214, 198, 198, 0	: REM M
8530 DATA	198, 230, 246, 222, 206, 198, 198, 0	: REM N
8540 DATA	56, 108, 198, 198, 198, 108, 56, 0	: REM O
8550 DATA	252, 102, 102, 120, 96, 96, 240, 0	: REM P
8560 DATA	56, 108, 198, 198, 218, 204, 118, 0	: REM Q
8570 DATA	252, 102, 102, 124, 108, 102, 226, 0	: REM R
8580 DATA	60, 102, 96, 60, 6, 102, 60, 0	: REM S
8590 DATA	126, 90, 24, 24, 24, 24, 60, 0	: REM T
8600 DATA	102, 102, 102, 102, 102, 102, 60, 0	: REM U
8610 DATA	102, 102, 102, 102, 102, 60, 24, 0	: REM V
8620 DATA	198, 198, 198, 214, 254, 238, 198, 0	: REM W
8630 DATA	198, 108, 56, 56, 108, 198, 198, 0	: REM X
8640 DATA	102, 102, 102, 60, 24, 24, 60, 0	: REM Y
8650 DATA	254, 198, 140, 24, 50, 102, 254, 0	: REM Z
8660 DATA	60, 48, 48, 48, 48, 48, 60, 0	: REM [
8670 DATA	192, 96, 48, 24, 12, 6, 3, 1	: REM \
8680 DATA	60, 12, 12, 12, 12, 12, 60, 0	: REM]
8690 DATA	24, 60, 126, 24, 24, 24, 24, 0	: REM ^
8700 DATA	0, 0, 0, 0, 0, 0, 0, 255	: REM _
8710 DATA	0, 48, 24, 12, 0, 0, 0, 0	: REM '
8720 DATA	0, 0, 120, 12, 124, 204, 118, 0	: REM a
8730 DATA	224, 224, 124, 102, 102, 102, 188, 0	: REM b
8740 DATA	0, 0, 60, 102, 96, 102, 60, 0	: REM c
8750 DATA	28, 12, 124, 204, 204, 204, 118, 0	: REM d
8760 DATA	0, 0, 60, 102, 126, 96, 60, 0	: REM e
8770 DATA	28, 54, 48, 120, 48, 48, 120, 0	: REM f
8780 DATA	0, 0, 62, 102, 102, 62, 6, 124	: REM g
8790 DATA	224, 96, 108, 118, 102, 102, 230, 0	: REM h
8800 DATA	24, 0, 56, 24, 24, 24, 60, 0	: REM i
8810 DATA	2, 0, 14, 6, 6, 102, 102, 60	: REM j
8820 DATA	224, 96, 102, 108, 120, 108, 230, 0	: REM k
8830 DATA	56, 24, 24, 24, 24, 24, 60, 0	: REM l
8840 DATA	0, 0, 108, 254, 214, 214, 198, 0	: REM m
8850 DATA	0, 0, 216, 102, 102, 102, 102, 0	: REM n
8860 DATA	0, 0, 60, 102, 102, 102, 60, 0	: REM o
8870 DATA	0, 0, 220, 102, 102, 124, 96, 240	: REM p
8880 DATA	0, 0, 118, 204, 204, 124, 12, 30	: REM q
8890 DATA	0, 0, 216, 108, 96, 96, 240, 0	: REM r
8900 DATA	0, 0, 60, 96, 60, 6, 124, 0	: REM s
8910 DATA	48, 48, 124, 48, 48, 54, 28, 0	: REM t

```

8920 DATA 0,0,102,102,102,102,62,0      :REM u
8930 DATA 0,0,102,102,102,60,24,0        :REM v
8940 DATA 0,0,198,214,214,254,108,0      :REM w
8950 DATA 0,0,198,108,56,108,198,0       :REM x
8960 DATA 0,0,102,102,102,62,6,124       :REM y
8970 DATA 0,0,126,76,24,48,126,0         :REM z
8980 DATA 14,24,24,112,24,24,14,0        :REM {
8990 DATA 24,24,24,24,24,24,24,0         :REM |
9000 DATA 112,24,24,14,24,24,112,0       :REM }
9010 DATA 118,216,0,0,0,0,0,0           :REM ~
9020 DATA 204,51,204,51,204,51,204,51    :REM

```

El siguiente programa nos permite experimentar con este último programa a nuestro gusto.

```

100 REM *****
110 REM * PROGRAMA DEMO PARA EJECUTAR *
120 REM * LA RUTINA DE LETRAS GRANDES *
130 REM *****
140 REM
150 CLS
160 PRINT "CARGANDO DATAS."
170 PRINT "ESPERE UN MOMENTO."
180 GOSUB 6000
190 CLS
200 INPUT "MENSAJE = ";A$
210 INPUT "COLUMNA = ";X
220 INPUT "FILA = ";Y
230 PRINT
240 PRINT "TIPO DE IMPRESION"
250 PRINT
260 PRINT "1 = IMPRESION NORMAL"
270 PRINT "2 = IMPRESION INVERTIDA"
280 PRINT "3 = NORMAL DE UN CARACTER"
290 PRINT "4 = INVERTIDA DE UN CARACTER"
300 PRINT "5 = NORMAL CADA LETRA SU CARACTER"
310 PRINT "6 = INVERTIDA CADA LETRA SU CARACTER"
320 PRINT
330 INPUT "TIPO = ";SW
331 IF SW=3 OR SW=4 THEN INPUT "CARACTER= ";M$:LET M$=LEFT$(" "+M$,1)
340 PRINT
350 PRINT "PULSA UNA TECLA PARA EMPEZAR"
360 LET B$=INKEY$
370 IF B$="" THEN GOTO 360
380 CLS
390 GOSUB 7000
400 PRINT
410 PRINT "PULSA UNA TECLA PARA"
420 PRINT "INTRODUCIR OTRO MENSAJE"
430 LET B$=INKEY$
440 IF B$="" THEN GOTO 430
450 GOTO 190

```

Estos dos últimos programas no se van a explicar debido al gran parecido a todo lo que hemos visto en estos dos últimos tomos. Las modificaciones al segundo programa para ordenadores distintos del IBM son:

COMMODORE:

```

7030 IF SW<2 THEN LET M$=CHR$(166):LET W$="- "
7040 IF SW=2 THEN LET M$="- ":LET W$=CHR$(166)
7140 POKE 214,Y+J-1:POKE 211,X
7340 IF X>40 THEN LET X=1:LET Y=Y+8

```

TRUCOS Y RUTINAS BASICAS

SPECTRUM:

```
7030 IF SW<2 THEN LET M$=CHR$(143):LET W$=" "  
7040 IF SW=2 THEN LET M$=" ":LET W$=CHR$(143)  
7080 LET N=CODE(A$(I))  
7100 IF SW=5 THEN LET M$=A$(I):LET W$=" "  
7110 IF SW=6 THEN LET W$=A$(I):LET M$=" "  
7140 PRINT AT Y+J-1,X;  
7340 IF X>32 THEN LET X=1:LET Y=Y+8
```

MSX:

```
7140 LOCATE X,Y+J-1  
7340 IF X>40 THEN LET X=1:LET Y=Y+8
```

AMSTRAD:

```
7140 LOCATE X,Y+J-1
```

Si no se utiliza el modo 2 de pantalla para ejecutar esta rutina hay que variar las siguientes líneas:

```
7340 IF X>40 THEN X=1:Y=Y+8:REM PANTALLA EN  
MOD0 1  
7340 IF X>32 THEN X=1:Y=Y+8:REM PANTALLA EN  
MOD0 0
```

Las variaciones al tercer programa son las siguientes:

COMMODORE:

```
150 PRINT CHR$(147)  
190 PRINT CHR$(147)  
360 GET B$  
380 PRINT CHR$(147)  
430 GET B$
```

SPECTRUM:

```
331 IF SW=3 OR SW=4 THEN INPUT "CARACTER=  
";M$:LET M$=(""+M$) (1)
```

■ Muestreador/sintetizador de señales de audiofrecuencia



IGUIENDO con el diseño de sistemas para el análisis y síntesis de señales de audiofrecuencia mediante ordenador personal, vamos a describir un circuito que permite el tratamiento digital de las señales y su manipulación por programa.

Los equipos electrónicos que acompañan a la mayoría de los cantantes disponen de un instrumento que utiliza el almacenamiento de la señal de forma digital. Permite la reproducción controlada del sonido producido por un instrumento real o por la voz, durante un período corto de tiempo, variando las características de repetición, intensidad sonora, modulación y cambio de la frecuencia fundamental, conservando el timbre original. Así pueden producirse sonidos muy parecidos al

de una trompeta desde un teclado y además simular la actuación de varios instrumentos simultáneos, accionados por un solo intérprete.

Todos los efectos mencionados pueden ser producidos mediante un equipo que se denomina muestreador ("sampler" en inglés) y que tiene por función principal la adquisición de una señal mediante muestras digitalizadas y almacenadas en una memoria. Estas muestras pueden convertirse otra vez en su valor analógico por el proceso inverso al de la adquisición, con la frecuencia de salida que convenga.

Vamos a describir los bloques principales de un equipo que permita producir los efectos indicados y su conexión a un ordenador personal. El equipo será de aplicación real, aunque con las limitaciones propias de los componentes utilizados. Lo importante es comprender el funcionamiento del sistema y disfrutar con la manipulación por programa de las señales, produciendo efectos que serían muy difíciles de crear incluso con los sintetizadores más costosos.



Fig. 1. Esquema de bloques del muestreador-sintetizador.

■ Esquema de bloques

La señal es amplificada mediante un amplificador de entrada, para que presente un nivel adecuado al conversor. La señal de entrada será generalmente alterna, por lo que se la desplaza a un nivel de continua correspondiente al valor medio de la máxima amplitud admisible por el conversor.

La señal amplificada se muestrea y el valor de la muestra podría ser mantenido durante el período de conversión por un circuito de muestra y mantenimiento. En la versión que se presenta no se ha incluido circuito de mantenimiento, por considerarse variaciones lentas de la señal de entrada para las aplicaciones reales.

El conversor, activado por el pulso de arranque genera un código digital por el valor de la muestra. Al finalizar la conversión presenta a la salida los ocho bits correspondientes al nivel de la señal.

El código se almacena en un registro controlado por el pulso de fin de conversión, que además indicará al ordenador que hay un nuevo dato disponible.

El ordenador, que estará observando el indicador de fin de conversión, almacena en la memoria el dato, llevando la cuenta de la posición.

El proceso se repite tantas veces como sea necesario para almacenar un número suficiente de muestras que hagan significativa la representación de la señal de entrada.

Una vez almacenada la señal en la memoria del ordenador, podemos realizar sobre los datos que la representan las operaciones adecuadas para su interpretación o modificación convenientes. En el caso más simple podremos sacarla de forma que reproduzca a la señal original en la parte siguiente del sistema. El contenido de la memoria puede sacarse al puerto de salida, que tendrá conectado un conversor digital analógico. Al cargar el registro con los diferentes valores se producirá una señal reproducción de la señal adquirida, pero que puede generarse a diferente frecuencia de la que se empleó para adquirirla.

El conversor digital/analógico presentará un nivel de continua correspondiente al valor almacenado y por tanto con un desplazamiento en continua. Mediante un filtro pasabanda, eliminaremos la componente continua y las componentes de alta frecuencia que

como consecuencia del muestreo discreto se presentarán a la salida.

El circuito mezclador de salida permite sumar la señal original y la procesada a través del ordenador, asignando a cada una de las señales el coeficiente deseado mediante potenciómetros. Con esto podremos añadir el efecto de ecos o de resonancias, mediante programas adecuados.

■ Amplificador de entrada

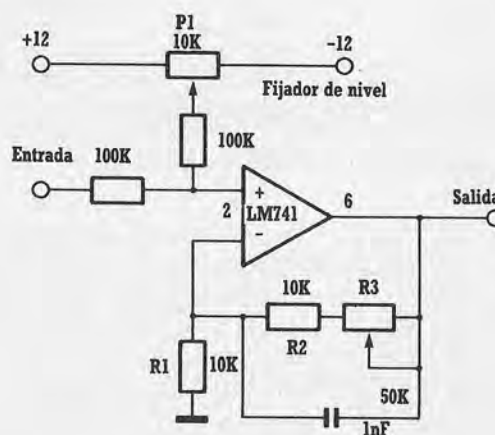


Fig. 2. Amplificador de entrada.

Se emplea un amplificador operacional típico, como los que hemos empleado en otros muchos circuitos anteriores. La ganancia se determina mediante las resistencias R1 y R2, por lo que esta segunda se compondrá de una parte fija y otra variable, para poder ajustar a los diferentes micrófonos.

El nivel de continua a la entrada se ajusta mediante el potenciómetro P1 para que a la salida, en reposo, haya un nivel de 2,5 voltios.

El condensador colocado en paralelo con la resistencia de realimentación sirve para limitar el ancho de banda de la señal de entrada. Podríamos incluso utilizar una etapa de filtro paso-bajo, para reducir los efectos de alias y que ya se han descrito en tomos anteriores. Como en este caso pretendemos producir efectos sonoros, no nos preocuparemos por ello, pero es conveniente recordarlo pues el efecto no es despreciable. Por otra parte, como el contenido en las frecuencias superiores es siempre bajo, los alias que aparecen son de baja amplitud, si es que no son eliminadas las componentes superiores, mediante el filtro.

■ Conversor analógico/digital

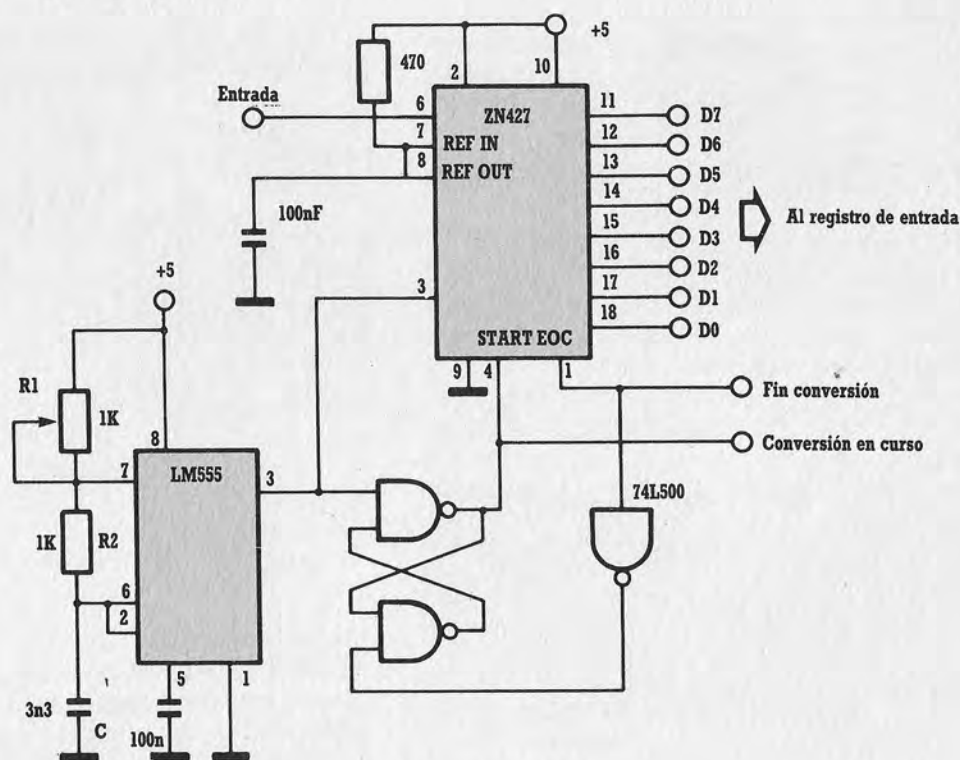


Fig. 3. Conversor analógico/digital.

Se utiliza uno más rápido que los empleados en tomos anteriores, con objeto de que se pueda cubrir un amplio ancho de banda de señales. El ZN427 puede trabajar con reloj de hasta 600 KHz. En nuestro ejemplo, para cubrir las frecuencias de audio, trabajaremos a 150 KHz. El periodo de muestreo será de unos 53 microsegundos. La frecuencia de muestreo la mantendremos en principio fija, jugando con la frecuencia de salida, mediante programa. Podemos añadir una circuitería de control de conversión que permita el control totalmente por programa.

El circuito tiene su propia señal de referencia, que se utiliza realizando un puente entre las entradas 7 y 8. Conviene que la referencia sea lo más estable e independiente del ruido externo, por lo que se conecta un condensador de filtro para reducirlo en lo posible. El método que se utiliza es de aproximaciones sucesivas.

El reloj de referencia se genera mediante un LM555, que ya hemos utilizado en otros circuitos. Se puede ajustar la frecuencia mediante una de las resistencias que definen

el periodo de oscilación, de acuerdo con la fórmula:

$$F = 1,45 / (C \times (R1 + 2 \times R2))$$

La conversión funciona de manera continua, activándose la señal de comienzo con la salida de un FF montado con puertas NAND cruzadas. El FF se desactiva con el pulso de final de conversión EOC. Esta señal está también disponible para la interfaz.

Se podría hacer la conversión controlada por programa haciendo que la entrada START estuviera conectada a una salida de decodificador de dirección. Simplemente con direccionar se comenzaría la conversión.

■ Registro de entrada al ordenador

Es un circuito similar a los puertos descritos en la tarjeta de ampliación de puerto, pero en este caso es un puerto de entrada con registro activado por el circuito exterior y leído por el ordenador cuando recibe la indicación de que está lleno. Se realiza con un registro de tipo 74LS374, con activación de la car-

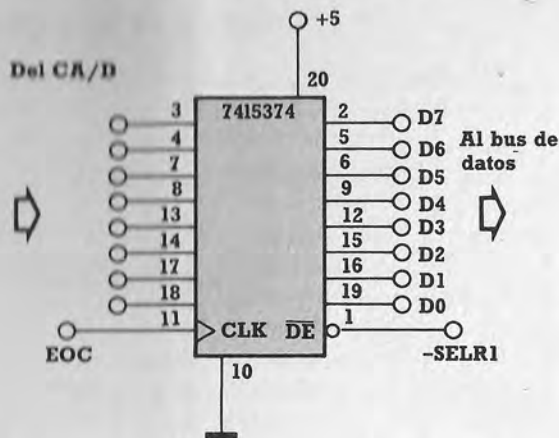


Fig. 4. Registro de entrada.

ga con el pulso de fin de conversión (EOC), pues requiere un pulso positivo. La lectura del registro se hace desde la CPU mediante una sentencia de lectura de puerto, decodificado como SELR1, por ejemplo. La decodificación actúa sobre la entrada de OE, que activa el circuito de tres estados de la salida.

Registro de salida

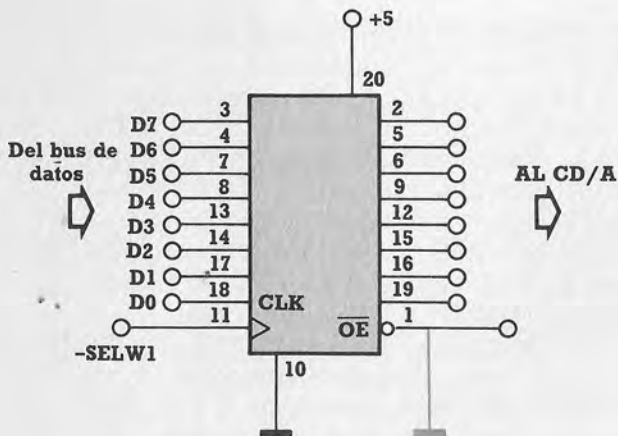


Fig. 5. Registro de salida.

Es exactamente un puerto de salida como los descritos en la tarjeta de ampliación de puertos. Se carga por programa, al generar su dirección el decodificador de direcciones de salida, SELW1, por ejemplo. La salida del registro está siempre activada, pues el control de OE se ha conectado a masa.

Se utiliza un registro de tipo 74LS374.

Conversor digital/análogo

Se utiliza un circuito como los empleados en la tarjeta de interfaces analógicas. El

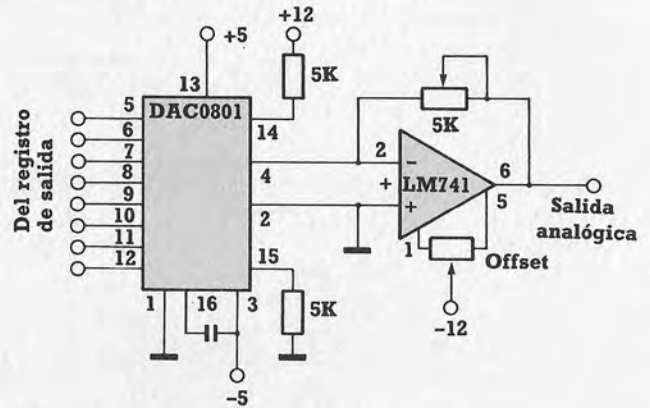


Fig. 6. Conversor digital/análogo.

circuito integrado más apropiado es el ADC0801, pero existen muchos otros de similares características. Realiza la conversión digital/análoga en modo corriente, por lo que es necesario añadir a la salida un amplificador que convierta la corriente en tensión y además determine la ganancia. Se ajusta la ganancia mediante la resistencia de 5K del bucle de realimentación. Es importante el ajuste del offset en este circuito para que no se sature en ninguno de los dos extremos.

Filtro de salida

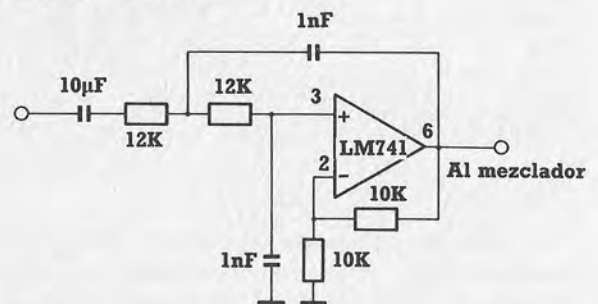


Fig. 7. Filtro de salida.

Se filtra la señal mediante un filtro de paso bajo, además de eliminar la componente de continua introducida por el desplazamiento necesario del convertor A/D, con un condensador en serie con la entrada. El filtro emplea un circuito integrado LM741, como en el resto de los amplificadores.

Mezclador

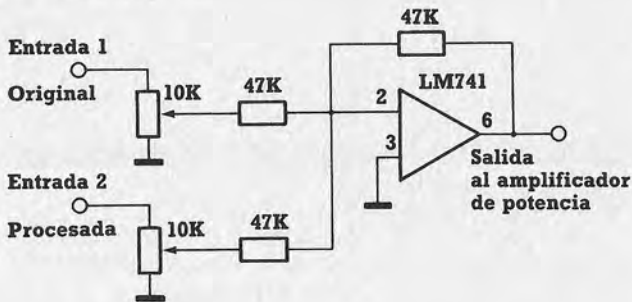


Fig. 8. Mezclador.

Simplemente permite sumar las señales original y procesada, mediante potenciómetros para producir el efecto conveniente.

La salida se llevará a un amplificador de potencia para actuar sobre un altavoz o a un conector para conexión de auriculares.

Alimentaciones

Todos los circuitos digitales mostrados se alimentan con 5 voltios. Los circuitos analógicos, como son amplificadores operacionales y conversores A/D y D/A se alimentan con tensiones de 12 voltios, positiva y negativa. Hay además necesidad de alimentación de -5 voltios para el C D/A, que por su bajo consumo puede generarse con diodo Zener a partir de la tensión de -12 voltios.

Como en todo circuito que amplifica señales analógicas de bajo nivel es conveniente disponer de pistas de alimentación y masa separadas para las alimentaciones de los circuitos digitales y desacoplar cada circuito con condensadores de 100 nF.

Programas de control

Para hacer funcionar el equipo así montado es necesario que el ordenador realice las funciones de adquisición, proceso y salida de las señales en la secuencia conveniente para cada caso. Vamos a ver los casos más simples y que convenientemente combinados convertirán a nuestro equipo en un muestreador-sintetizador programable. Los programas han de realizarse en lenguaje de máquina o en un lenguaje compilable, pues si no es así, la frecuen-

cia de toma de muestras resultará muy baja para las señales acústicas. Los ejemplos que se mencionan serán necesarios para que resulten prácticos.

Muestreo y salida retardada

El primer caso simple consiste en muestrear la señal y sacarla con un tiempo de retraso con relación a la señal de entrada. La adquisición de datos se activa por el programa y según el retardo deseado, se almacenan las muestras durante un período y se presentan a la salida sin modificación. Es decir, la muestra de entrada se almacena en la posición X y se presenta a la salida la muestra existente en la posición X-R, siendo R el número de muestras de retardo. Expresado en tiempo el retardo será R/N , siendo N el número de muestras por segundo que se estén tomando. El almacenamiento ha de ser, por supuesto, de tipo circular, con un tamaño mínimo correspondiente a R. Se realiza el acceso de forma circular mediante el cálculo del resto módulo del número de elementos de la tabla.

```

10 REM *****
20 REM PROGRAMA DE TRATAMIENTO DE SEÑALES
30 REM *****
40 REM UTILIZA LA TARJETA C A/D Y C D/A
50 REM EN EL 10M PL
60 REM *****
70 DEFINIR A-Z
80 DIM B(32000) REM VECTOR DE MUESTRAS
90 REM DESCRIPCION DEL PUERTO DE ENTRADA
91 REM SELECCIONAR DEL C A/D
94 REM DESCRIPCION DEL PUERTO DE ESTADO
95 REM ELEGIR OBTENEDOR FUENTE
96 REM DESCRIPCION DEL PUERTO DE SALIDA
97 REM SELECCIONAR DEL C D/A
100 REM DEFINICION DE PULSOS
110 PULSOS=(40/44) REM DATOS DEL C A/D Y D/A
130 PULSOS=0 REM DATOS DE ESTADO Y CONTROL
200 RLY OFF
210 CLR
220 PRINT " PROGRAMA DE TRATAMIENTO DE SEÑALES"
230 PRINT
240 PRINT " VERIFICAR LAS CONEXIONES DE ENTRADA"
250 PRINT
260 PRINT " SELECCIONAR FUENTE"
270 PRINT
280 PRINT " 1- CON LOS SEÑALES Y SIN RETARDO"
290 PRINT
300 PRINT " 2- MUESTREO DE RETARDO SUFICIENTEMENTE"
310 PRINT
320 PRINT " 3- MUESTREO"
330 PRINT
340 PRINT " 4- MUESTREO DE RETARDO"
350 IF SECCION 1 THEN GOTO 210
360 IF SECCION 2 THEN GOTO 210
370 IF SECCION 3 THEN GOTO 210
380 IF SECCION 4 THEN GOTO 210
390 GOTO 210
400 GOTO 210
410 GOTO 210
420 GOTO 210
430 GOTO 210
440 GOTO 210
450 GOTO 210
460 GOTO 210
470 GOTO 210
480 GOTO 210
490 GOTO 210
500 GOTO 210
510 GOTO 210
520 GOTO 210
530 GOTO 210
540 GOTO 210
550 GOTO 210
560 GOTO 210
570 GOTO 210
580 GOTO 210
590 GOTO 210
600 GOTO 210
610 GOTO 210
620 GOTO 210
630 GOTO 210
640 GOTO 210
650 GOTO 210
660 GOTO 210
670 GOTO 210
680 GOTO 210
690 GOTO 210
700 GOTO 210
710 GOTO 210
720 GOTO 210
730 GOTO 210
740 GOTO 210
750 GOTO 210
760 GOTO 210
770 GOTO 210
780 GOTO 210
790 GOTO 210
800 GOTO 210
810 GOTO 210
820 GOTO 210
830 GOTO 210
840 GOTO 210
850 GOTO 210
860 GOTO 210
870 GOTO 210
880 GOTO 210
890 GOTO 210
900 GOTO 210
910 GOTO 210
920 GOTO 210
930 GOTO 210
940 GOTO 210
950 GOTO 210
960 GOTO 210
970 GOTO 210
980 GOTO 210
990 GOTO 210

```

```

1020 INPUT "NUMERO MAXIMO DE REPETICIONES";MAXIMO
1100 INPUT "PULSE ENTER CUANDO ESTE PREPARADO";X$
1140 PRINT
1150 PRINT "CAPTANDO SEÑALES"
1190 FOR J=1 TO MAXIMO
1200   FOR I=1 TO 320
1210     A=INP(PSTAT)
1220     IF 0=(A AND 1) THEN 1160
1230     B(I)=INP(PDATOS)
1240     OUT PDATOS,B((I+R) MOD 320)
1250   NEXT I
1260 NEXT J
1270 RETURN
1990 REM * * * * PRESENTAR RESULTADO * * * *
2000 CLS
2010 PRINT "LISTADO DE LA SEÑAL CAPTADA"
2020 PRINT
2030 FOR I=1 TO 320
2040   PRINT " ";HEX$(B(I));
2050 NEXT I
2060 RETURN
3000 REM TERMINA
3010 OUT PSTAT,0
3100 RETURN

```

■ Muestreo y salida modulada

Es un caso parecido al anterior, pero en el que la señal se multiplica por un coeficiente variable con el tiempo, con lo que presentará una variación como la de trémolo. Variando la frecuencia y la amplitud de la modulación podremos generar diferentes efectos.

Sobre el ejemplo presentado se modificará la salida de la señal muestreada multiplicándola por un factor dependiente de la posición a leer. Si además se modifica la frecuencia de modulación, se incluirá en el bucle de lectura de la señal.

■ Generación de señales arbitrarias

Mediante la asignación de niveles arbitrarios para cada nivel de entrada pueden generarse señales que, siendo arbitrarias parecen tener una sistematicidad.

Para producirlas se sustituirá el valor captado por otro almacenado en una tabla. Es decir, el valor a sacar no es el que se almacenó en el vector de señales sino el que se encuentre almacenado en la tabla, en la posición correspondiente al valor de entrada. La tabla se generará de acuerdo con la ley que se desea ensayar, evitando el realizar cálculos durante el proceso de búsqueda en la tabla, pues resultaría lento. Sobre el ejemplo anterior será necesario añadir la generación o lectura de la tabla de conversión y en la sentencia OUT al puerto de salida, sustituir la referencia al vec-

tor de señales por la del acceso indexado a la tabla.

■ Muestreo y generación de armónicos

La señal se muestreará como en los casos anteriores pero por un tiempo determinado. Una vez lleno el espacio reservado y que corresponderá a un tiempo adecuado al efecto a producir se detendrá la adquisición de muestras. El ordenador quedará en disposición de pulsar las teclas, generándose una señal igual a la muestreada pero con una frecuencia de salida dependiente de cada tecla, por lo que se producirán diferentes tonos con el mismo timbre. En el equipo que hemos diseñado solamente podrán producirse sonidos monofónicos pues sólo disponemos de un circuito de entrada y otro de salida. Los equipos profesionales suelen disponer de más de uno para poder producir señales polifónicas, pero como se comprenderá a un coste mucho más elevado.

Para realizarlo prácticamente, sobre el programa ejemplo mostrado se realizará una vez la toma de muestra, sincronizándola con la entrada, es decir, eliminando las muestras iniciales que sean inferiores a un determinado nivel. Después se generará una reproducción de la señal completa, cada vez que se pulse una tecla, variando el tiempo entre la salida de dos muestras. Es necesario calibrar en cada caso, pues al ser muy dependientes del reloj de cada máquina, no se puede predecir la frecuencia que puede resultar.

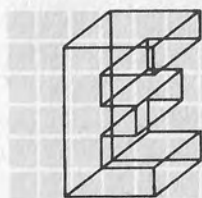
■ Conclusión

Se ha presentado un circuito de adquisición y generación de señales analógicas que, controlado por nuestro ordenador personal, permite la creación de efectos sonoros interesantes. El mismo circuito permite el almacenamiento de la información relacionada con las señales de forma digital y mediante programa pueden realizarse operaciones de filtrado y reconstrucción de la señal de entrada. En el próximo tomo veremos cómo procesar las muestras con otro punto de vista: conocer su contenido frecuencial.

APRENDER CON EL ORDENADOR

MATEMATICAS

■ Resolución de un sistema de tres ecuaciones por el método de Cramer



STE método de resolución de ecuaciones se basa en el cálculo de determinantes. Lo proponemos para resolver un sistema de tres ecuaciones con tres incógnitas, pero puede generalizarse a n . Para ello debe utilizarse

el programa de resolución de determinantes que vimos en otro fascículo.

Partimos del siguiente sistema de ecuaciones:

$$\begin{aligned}a_1 x + b_1 y + c_1 z &= d_1 \\a_2 x + b_2 y + c_2 z &= d_2 \\a_3 x + b_3 y + c_3 z &= d_3\end{aligned}$$

La solución de este sistema será:

$$\begin{aligned}x &= \frac{\begin{vmatrix} d_1 & b_1 & c_1 \\ d_2 & b_2 & c_2 \\ d_3 & b_3 & c_3 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}} \\y &= \frac{\begin{vmatrix} a_1 & d_1 & c_1 \\ a_2 & d_2 & c_2 \\ a_3 & d_3 & c_3 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}} \\z &= \frac{\begin{vmatrix} a_1 & b_1 & d_1 \\ a_2 & b_2 & d_2 \\ a_3 & b_3 & d_3 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}}\end{aligned}$$

Es en definitiva lo que se ha aplicado para la realización del programa.

```
10 CLS
20 PRINT "*****"
30 PRINT "*** RESOLUCION DE SISTEMAS DE 3 ECUACIONES ***"
40 PRINT "***** POR EL METODO DE CRAMER *****"
50 PRINT "*****"
60 PRINT:PRINT:PRINT
70 DIM CO(3,3):DIM T(3)
80 REM *****
90 REM ** INTRODUCCION DE DATOS **
100 REM *****
110 FOR I=1 TO 3
120 PRINT "INTRODUCE LOS COEFICIENTES DE LA ECUACION ":I
130 FOR J=1 TO 3
140 INPUT CO(I,J)
150 NEXT J
160 INPUT "TERMINO INDEPENDIENTE=":T(I)
170 NEXT I
180 REM *****
190 REM ** PRESENTACION EN PANTALLA **
200 REM *****
210 CLS
220 PRINT "RESOLUCION DEL SISTEMA DE 3 ECUACIONES:"
```

```

230 FOR I=1 TO 3
240 PRINT CO(I,1):"X=":CO(I,2):"Y=":CO(I,3):"Z=":T(I)
250 NEXT I
260 PRINT:PRINT
270 REM *****
280 REM ** RESOLUCION **
290 REM *****
300 DIM DET(3,3)
310 FOR I=1 TO 3
320 FOR J=1 TO 3
330 LET DET(I,J)=CO(I,J)
340 NEXT J
350 NEXT I
360 GOSUB 560
370 LET DN=D
380 PRINT :PRINT :PRINT :PRINT
390 REM *****
400 REM ** INCOGNITAS **
410 REM *****
420 FOR A=1 TO 3
430 FOR I=1 TO 3
440 FOR J=1 TO 3
450 LET DET(I,J)=CO(I,J)
460 IF A=J THEN LET DET(I,J)=T(I)
470 NEXT J
480 NEXT I
490 GOSUB 560
500 LET NU=D
510 IF A=1 THEN PRINT "X=":NU/DN
520 IF A=2 THEN PRINT "Y=":NU/DN
530 IF A=3 THEN PRINT "Z=":NU/DN
540 NEXT A
550 END
560 REM *****
570 REM ** SUBROUTINA DETERMINANTE **
580 REM *****
590 D=DET(1,1)*DET(2,2)*DET(3,3)-DET(2,3)*DET(3,2)-DET(1,2)*(DET(2,1)*DET(3,3)-
-DET(2,3)*DET(3,1))+DET(1,3)*(DET(2,1)*DET(3,2)-DET(2,2)*DET(3,1))
600 RETURN

```

Modificaciones para otros ordenadores

COMMODORE:

10 PRINT CHR\$(147)

SPECTRUM:

550 GOTO 9999

Comprobando el funcionamiento

Vamos a intentar resolver la siguiente ecuación:

$$\begin{aligned} 3x + 2y - z &= -1 \\ 2x - 3y + 2z &= 3 \\ x + y + z &= 4 \end{aligned}$$

Aplicamos el método de Cramer, el mismo que utiliza el programa en el ordenador:

$$x = \frac{\begin{vmatrix} -1 & 2 & -1 \\ 3 & -3 & 2 \\ 4 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} 3 & 2 & -1 \\ 2 & -3 & 2 \\ 1 & 1 & 1 \end{vmatrix}} = 0$$

$$y = \frac{\begin{vmatrix} 1 & 1 & 1 \\ 3 & -1 & -1 \\ 2 & 3 & 2 \end{vmatrix}}{\begin{vmatrix} 3 & 2 & -1 \\ 2 & -3 & 2 \\ 1 & 1 & 1 \end{vmatrix}} = 1$$

$$z = \frac{\begin{vmatrix} 3 & 2 & -1 \\ 2 & -3 & 3 \\ 1 & 1 & 4 \end{vmatrix}}{\begin{vmatrix} 3 & 2 & -1 \\ 2 & -3 & 2 \\ 1 & 1 & 1 \end{vmatrix}} = 3$$

Comprueba con tu ordenador que los resultados son correctos.

NATURALEZA Y TECNOLOGIA

Botánica. Las hojas

Este es el segundo programa de botánica que presentamos; en esta ocasión el tema de estudio es la hoja.

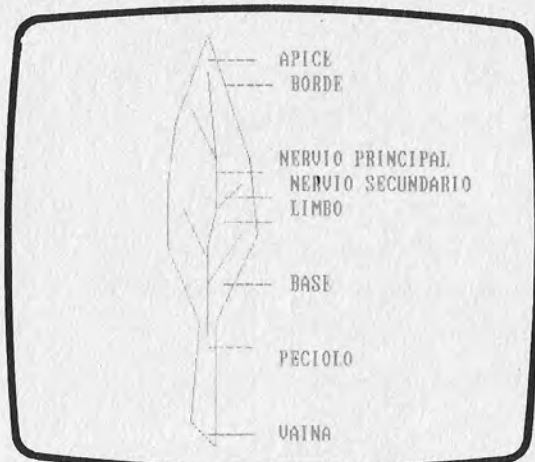
Las hojas son expansiones laminares de los tallos, que, cargadas de clorofila, se constituyen en los más importantes órganos fotosintéticos. Generalmente son planas y presentan simetría bilateral.

Las partes de la hoja son tres:

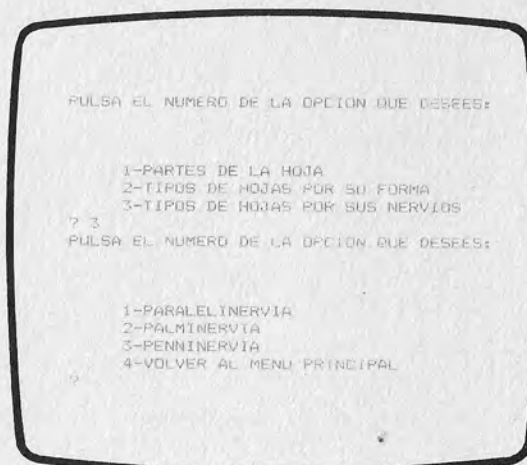
1. Vaina. Es la zona de la base de la hoja de forma tubular. Abraza al tallo de forma más o menos total.

2. Pecíolo. Es el rabillo que une la hoja al tallo. Las hojas que carecen de pecíolo se denominan sentadas.

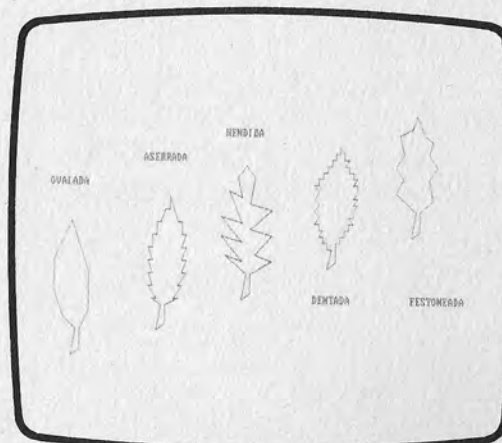
3. Limbo. Es la porción plana ensanchada. La parte superior se llama haz y la inferior envés. En esta parte sobresalen los nervios que son haces de vasos liberoleñosos por cuyo interior circula la savia.



Hemos creado un programa que parte de un menú en el que se puede elegir la opción deseada pulsando el número que la identifica.



Como puede observarse, ese menú puede llevarnos a otro posterior, de tal forma que obtenemos una lista de la clasificación deseada y, por último, podemos ver la opción elegida.



Pasamos ya a ver el programa:

```

10 REM *****
20 REM ** BOTANICA: LAS HOJAS **
30 REM *****
40 CLS
50 SCREEN 2
60 REM *****
70 REM ** LECTURA DE DATAS EN M **
80 REM *****
90 LET N=488
100 DIM M(N)
110 FOR I=1 TO N
120 READ M(I)
130 NEXT I
140 REM *****
150 REM ** MENU DE OPCIONES **
160 REM *****
170 PRINT "PULSA EL NUMERO DE LA OPCION QUE DESEES:"
180 PRINT :PRINT :PRINT
190 PRINT "      1-PARTES DE LA HOJA"
200 PRINT "      2-TIPOS DE HOJAS POR SU FORMA"
210 PRINT "      3-TIPOS DE HOJAS POR SUS NERVIOS"
220 INPUT R
230 IF R=1 THEN GOSUB 660

```

APRENDER CON EL ORDENADOR

```

240 IF R=2 THEN GOSUB 280
250 IF R=3 THEN GOSUB 490
260 CLS:GOTO 170
270 END
280 REM *****
290 REM ** MENU DE TIPOS DE HOJAS POR SU FORMA **
300 REM *****
310 CLS
320 PRINT "PULSA EL NUMERO DE LA OPCION QUE DESEES:"
330 PRINT :PRINT
340 PRINT "    1-OVALADA"
350 PRINT "    2-DENTADA"
360 PRINT "    3-ASERRADA"
370 PRINT "    4-FESTONEADA"
380 PRINT "    5-HENDIDA"
390 PRINT "    6-VOLVER AL MENU PRINCIPAL"
400 INPUT I
410 CLS
420 IF I=1 THEN LET A=55: GOSUB 870: PRINT "OVALADA":INPUT 0
430 IF I=2 THEN LET A=78: GOSUB 870: PRINT "DENTADA":INPUT 0
440 IF I=3 THEN LET A=199: GOSUB 870: PRINT "ASERRADA":INPUT 0
450 IF I=4 THEN LET A=272: GOSUB 870: PRINT "FESTONEADA":INPUT 0
460 IF I=5 THEN LET A=309: GOSUB 870: PRINT "HENDIDA":INPUT 0
470 IF I=6 THEN RETURN
480 GOTO 310
490 REM *****
500 REM ** MENU DE TIPOS DE HOJAS POR SUS NERVIOS **
510 REM *****
520 CLS
530 PRINT "PULSA EL NUMERO DE LA OPCION QUE DESEES:"
540 PRINT :PRINT
550 PRINT "    1-PARALELINERVIA"
560 PRINT "    2-PALMINERVIA"
570 PRINT "    3-PENNINERVIA"
580 PRINT "    4-VOLVER AL MENU PRINCIPAL"
590 INPUT I
600 CLS
610 IF I=1 THEN LET A=55 : GOSUB 870: LET A=354: GOSUB 870:PRINT "PARALELINERVIA":INPUT 0
620 IF I=2 THEN LET A=377: GOSUB 870: LET A=418: GOSUB 870:PRINT "PALMINERVIA":INPUT 0
630 IF I=3 THEN LET A=439: GOSUB 870: LET A=459: GOSUB 870:PRINT "PENNINERVIA":INPUT 0
640 IF I=4 THEN RETURN
650 GOTO 520
660 REM *****
670 REM ** RUTINA PARTES DE LA HOJA **
680 REM *****
690 CLS
700 LET A=1 : GOSUB 870
710 LET A=24 : GOSUB 870
720 FOR I=1 TO 8
730 READ A,B
740 PSET (A*7,300-B*7)
750 DRAW "M+40,+0"
760 NEXT I
770 LOCATE 6,14:PRINT "APICE"
780 LOCATE 7,15:PRINT "BORDE"
790 LOCATE 10,14:PRINT "NERVIO PRINCIPAL"
800 LOCATE 11,15:PRINT "NERVIO SECUNDARIO"
810 LOCATE 12,15:PRINT "LIMBO"
820 LOCATE 15,15:PRINT "BASE"
830 LOCATE 18,14:PRINT "PECIULO"
840 LOCATE 21,14:PRINT "VAINA"
850 INPUT I
860 RETURN
870 REM *****
880 REM ** RUTINA DE DIBUJO **
890 REM *****
900 PSET (M(A+1)*7,300-M(A+2)*7)
910 FOR I=A+3 TO A+3+M(A) STEP 2
920 LET X$=STR$(M(I)*7): LET Y=(-7)*M(I+1): LET Y$=STR$(Y)
930 LET DIB$="M+"+X$+"."+Y$
940 DRAW DIB$
950 NEXT I
960 RETURN
970 DATA 18,+6,+3,+1,+8,-4,+6,+0,+5,+1,+5,+4,+7,+5,-10,+1,-6,-5,-7,+0,-10,-3,+2
980 DATA 26,+8,+10,+0,+4,+4,+4,-4,-4,+0,+2,-3,+4,+3,-4,+0,+1,+1,+3,+3,+2,-3,-2,+0,+4,-3,+4,+3,-4,-1,+7
990 DATA 19,+10,+10,+1,+4,-3,+2,-2,+4,+0,+5,+4,+6,+4,-8,+0,-6,-2,-3,+0,-3,-2,-1
1000 DATA 117,+10,+10,+1,+3,-1,+0,+0,+1,-1,+0,+0,+1,-1,+0,+1,+1,-1,+0,+0,+1,-1,+

```

```

0,+1,+1,-1,+1,+1,+1,-1,+1,+1,+1,-1,+1,+1,+0,+0,+1,+1,+0,+0,+1,+1,+0,+0,+1,+1,+0,
+0,+1,+1,+0,+0,+0,+1,+1,+0,+0,+1,+1,-1
1010 DATA +0,-1,+1,+0,-1,-1,+1,+0,+0,-1,+1,+0,-1,-0,+1,+0,+0,-1,+1,+0,-1,-1,+1,+
0,+0,-1,-1,-1,+1,+0,-1,-1,+1,+0,-1,-1,+0,-1,-1,+0,+1,-1,-1,+0,+0,-1,-1,+0,+0,-1,
-1,+0,+0,-1,-1,+0,+0,-3,-2,-1
1020 DATA 69,+10,+10,+1,+4,-2,+1,+1,+0,-2,+2,+1,+0,-1,+2,+1,+0,-1,+2,+1,+0,-1,+2,
+1,+0,+0,+2,+1,+0,+0,+2,+1,+0,+1,+2,+1,+0,+0,+2,+2,-4,+1,+0,+0,-2,+1,+0,-1,-2,+
1,+0,-1,-2,+1,+0,-1,-2,+1,+0,-2,-2,+1,+0,-2,-2,+1,+0,-3,-1,+0,-3,-2,-1
1030 DATA 33,+10,+10,+1,+4,-5,+3,+2,+2,-2,+2,+2,+2,-1,+3,+3,+1,+1,+2,+2,-3,+2,-1,
-1,-2,+2,-2,-2,-2,+1,-3,-3,-2,+0,-3,-2,-1
1040 DATA 41,+10,+10,+1,+4,-5,+3,+4,-1,-5,+4,+5,-1,-5,+4,+5,-1,-3,+5,+3,-1,-1,+3,
+2,+2,+2,-2,-1,-4,+5,-1,-5,-3,+5,-1,-4,-3,+4,-1,-5,-2,+0,-3,-2,-1
1050 DATA 19,+10,+10,+2,+10,-2,+11,+0,-11,+1,-5,-2,+5,+0,+4,+1,+7,-3,-6,+1,-7,+3,
,-3
1060 DATA 37,+10,+10,+1,+5,-6,-2,-2,+3,+1,+2,+4,+0,-5,+5,+3,+2,+3,-4,+5,+5,+2,-7,
+2,+0,+3,-1,-1,-3,-4,-2,+1,-2,-2,-1,-3,+5,+0,-5,-2,+0
1070 DATA 17,+12,+16,-7,+0,+7,+0,-6,+6,+6,-6,+1,+8,-1,-8,+6,+2,-6,-2,+3,-3
1080 DATA 17,+10,+10,+0,+2,-7,+4,+1,+6,+6,+10,+8,-10,+1,-7,-7,-3,+0,-3,-2,+1
1090 DATA 25,+11,+13,+0,+2,-5,+3,+5,-3,+0,+3,+4,+2,-4,-2,+0,+2,-5,+3,+5,-3,+0,+2,
+3,+3,-3,-3,-1,+7
1100 DATA 8,2,8,9,10,14,10,19,10,21,9,23,10,30,8,32

```

Modificaciones para otros ordenadores

SPECTRUM:

```

50 no se pone
270 GOTO 9999
740 PLOT A,B
750 DRAW 20,0
770 PRINT AT 6,14;"APICE"
780 PRINT AT 7,15;"BORDE"
790 PRINT AT 10,14;"NERVIO PRINCIPAL"
800 PRINT AT 11,15;"NERVIO SECUNDARIO"
810 PRINT AT 12,15;"LIMBO"
820 PRINT AT 15,15;"BASE"
830 PRINT AT 18,14;"PECIOLO"
840 PRINT AT 21,14;"VAINA"
900 PLOT M(A+1),M(A+2)
920 no se pone
930 no se pone
940 DRAW M(I),M(I+1)

```

AMSTRAD:

```

50 MODE 2
740 PLOT A,B
750 DRAW 20,0
770 LOCATE 14,6: PRINT "APICE"
780 LOCATE 15,7: PRINT "BORDE"
790 LOCATE 14,10: PRINT "NERVIO PRINCIPAL"
800 LOCATE 15,11: PRINT "NERVIO SECUNDARIO"
810 LOCATE 15,12: PRINT "LIMBO"
830 LOCATE 14,18: PRINT "PECIOLO"
840 LOCATE 14,21:"VAINA"
900 PLOT M(A+1)*5,M(A+2)*5

```

```

920 no se pone
930 no se pone
940 DRAW M(I)*5,M(I+1)*5

```

SOCIEDAD

■ La Comunidad Económica Europea

La Europa occidental que comenzó a rehacerse con la ayuda de los Estados Unidos (Plan Marshall, 1947) vio nacer, para coordinar esos fondos, una primera organización (la O.E.C.E.: Organización Europea de Cooperación Económica, 1948) que sería el germen de los futuros propósitos integradores. A él debemos agregar un primer intento de carácter mucho más localizado como es el del Benelux (1944).

El éxito de la posterior Comunidad del Carbón y del Acero (la C.E.C.A.) movió a una ampliación de los objetivos. Así se llegó por el Tratado de Roma (1957) a la formación de la Comunidad Económica Europea (la C.E.E.), que tiene como miembros fundadores a seis países: Francia, Alemania Federal, Italia, Bélgica, Países Bajos y Luxemburgo.

Recientemente España ha pasado a ser miembro de la C.E.E., es algo que nosotros no debemos olvidar por las grandes cosas que ello conlleva para el país, por eso hemos pensado que debe ser uno de los temas más importantes entre nuestros programas y hemos querido que se tratara de un programa bastante completo.

```

10 CLS
20 PRINT "*****"
30 PRINT "** LA COMUNIDAD ECONOMICA EUROPEA **"
40 PRINT "C.E.E.      ***"
50 PRINT "*****"
60 PRINT :PRINT :PRINT
70 PRINT "      MENU DE OPCIONES:"
80 PRINT :PRINT
90 PRINT "      1-ORGANIGRAMA DE LA C.E.E."
100 PRINT "      2-HISTORIA CRONOLOGICA."
110 PRINT "      3-TEST DE CONOCIMIENTOS."
120 PRINT "      4-SALIR DEL PROGRAMA.  "
130 INPUT O
140 IF O=1 THEN GOSUB 1000
150 IF O=2 THEN GOSUB 2000
160 IF O=3 THEN GOSUB 3000
170 IF O=4 THEN END
180 GOTO 10
1000 REM *****
1010 REM ** RUTINA ORGANIGRAMA **
1020 REM *****
1030 CLS
1040 PRINT "      ORGANIGRAMA DE LA C.E.E."
1050 PRINT
1060 PRINT "      ..... "
1070 PRINT "      :      COMITE      : "
1080 PRINT "      ..:ECONOMICO Y SOCIAL:.. "
1090 PRINT "      : : organo consultivo: : "
1100 PRINT "      : :.....: : "
1110 PRINT "      : :      : "
1120 PRINT "      :.....: ..... "
1130 PRINT "      : COMISION :      : CONSEJO DE: "
1140 PRINT "      : EUROPEA :.....: MINISTROS : "
1150 PRINT "      :O.gestor v:      :O.ejecutivo: "
1160 PRINT "      :ejecutivo :..... : principal : "
1170 PRINT "      :.....: : :.....: "
1180 PRINT "      : :      : "
1190 PRINT "      : :      :..... "
1200 PRINT "      :.....: ..... "
1210 PRINT "      : PARLAMENTO :      : TRIBUNAL : "
1220 PRINT "      : EUROPEO :      : DE : "
1230 PRINT "      :O.para control: : JUSTICIA : "
1240 PRINT "      :de la comision: :O.judicial: "
1250 PRINT "      :.....: :.....: "
1260 INPUT O:RETURN
2000 REM *****
2010 REM ** RUTINA HISTORIA **
2020 REM *****
2030 CLS
2040 PRINT "      HISTORIA CRONOLOGICA "
2050 PRINT
2060 PRINT "      FECHA :      ACONTECIMIENTO"
2070 PRINT "      ..... "
2080 PRINT "      25/3/57 : TRATADOS DE ROMA surgen la C.E.E. y el EURATOM"
2090 PRINT "      1/7/68 : Supresión de barreras arancelarias entre los
      : países fundadores
2100 PRINT "      1/7/71 : Establecen preferencias arancelarias genera-
      : lizadas entre C.E.E. y el Tercer Mundo."
2110 PRINT "      1/1/73 : Ingresan REINO UNIDO. IRLANDA y DINAMARCA."
2120 PRINT "      5/6/74 : Referendum en Reino Unido sobre la permanencia
      : en la C.E.E. con resultado afirmativo."
2130 PRINT "      12/6/75 : Grecia solicita su ingreso en la C.E.E."
2140 PRINT "      1/4/76 : Entra en vigor el convenio de Lomé. "
2150 PRINT "      28/3/77 : Portugal solicita su ingreso en la C.E.E."
2160 PRINT "      1/7/77 : Entra en vigor la unión aduanera.
      : Abolición de aranceles industriales entre la C.E.E.
      : y la E.F.T.A."
2170 PRINT "      28/7/77 : España solicita su ingreso en la C.E.E."
2180 PRINT "      1/1/86 : España ingresa en la C.E.E."
2190 INPUT O: RETURN
3000 REM *****
3010 REM ** RUTINA TEST **
3020 REM *****
3030 CLS:LET C=0
3040 PRINT "      TEST DE CONOCIMIENTOS "
3050 FOR I=1 TO 10
3060 READ P$,R1$,R2$,R3$
3070 PRINT P$:PRINT
3080 PRINT "1-":R1$:PRINT
3090 PRINT "2-":R2$:PRINT
3100 PRINT "3-":R3$:PRINT
3110 PRINT

```

```

3120 INPUT R
3130 READ V
3140 IF R=V THEN C=C+1:PRINT "!!!CORRECTO!!!":GOTO 3170
3150 PRINT "NO ES CORRECTO"
3160 PRINT "LA CORRECTA ES LA ":V
3170 NEXT I
3180 PRINT :PRINT "HAS CONTESTADO CORRECTAMENTE ":C:" PREGUNTAS"
3190 INPUT Q
3200 RETURN
5000 DATA "¿Cuántos países forman la C.E.E.?", "11", "9", "12", 1
5010 DATA "¿Cuántos fueron los miembros fundadores?", "3", "6", "7", 2
5020 DATA "Los países fundadores fueron:", "Francia, Alemania Federal, Italia, Bélgica, Países Bajos y Luxemburgo", "Francia, Alemania, Irlanda, Italia, Bélgica, Luxemburgo y Suiza", "Italia, Francia, Rusia, Irlanda, Bélgica y Japón", 1
5030 DATA "La C.E.E. surge en:", "El Tratado de París", "Los Tratados de Berlín", "Los Tratados de Roma", 3
5040 DATA "Grecia ingresó en la C.E.E. en:", "1980", "1975", "1972", 2
5050 DATA "España pidió el ingreso en la C.E.E. en:", "1980", "1977", "1978", 2
5060 DATA "España ingresó en la C.E.E. en:", "1986", "1985", "1977", 1
5070 DATA "El Comité Económico y Social es:", "El Órgano judicial", "El Órgano ejecutivo", "El Órgano consultivo", 3
5080 DATA "El Órgano ejecutivo principal está formado por:", "El Tribunal de Justicia", "El Parlamento Europeo", "El Consejo de Ministros", 3
5090 DATA "El Órgano judicial está formado por:", "El Tribunal de Justicia", "El Parlamento Europeo", "El Consejo de Ministros", 1

```

El programa presenta un menú en pantalla con varias opciones:

```

*****
** LA COMUNIDAD ECONOMICA EUROPEA **
** C.E.E. **
*****

MENU DE OPCIONES:

1-ORGANIGRAMA DE LA C.E.E.
2-HISTORIA CRONOLOGICA.
3-TEST DE CONOCIMIENTOS.
4-SALIR DEL PROGRAMA.

```

```

ORGANIGRAMA DE LA C.E.E.
.....
: COMITE :
...ECONOMICO Y SOCIAL:...
: : organo consultivo: :
: : : :
: : : :
.....
: COMISION : : CONSEJO DE:
: EUROPEA : : MINISTROS :
: O.gestor v: : O.ejecutivo:
: ejecutivo : : : :
: : : :
: : : :
.....
: PARLAMENTO : : TRIBUNAL :
: EUROPEO : : DE :
: O.para control: : JUSTICIA :
: de la comision: : O.judicial:
: : : :

```

La primera de ellas permite ver un organigrama de la distribución de los organismos en la C.E.E. El programa permite mantenerlo en pantalla para aprenderlo hasta que se desee y pulsando una tecla se vuelve al menú principal.

La segunda opción consiste en una lista de fechas y acontecimientos importantes en la historia de la Comunidad, es importante conocer algunas de ellas, de esta forma se aprende a relacionar hechos históricos.

La tercera opción consiste en un test, muy necesario para poder evaluar los conocimientos que tenemos sobre el tema.

Variaciones para otros ordenadores

El programa ha sido realizado en un BASIC muy general, por lo que puede ser utilizado en todos los ordenadores con sólo tener en cuenta lo siguiente:

COMMODORE:

Cambiar las sentencias CLS por PRINT CHR\$(147)

SPECTRUM:

Cambiar la sentencia END por GOTO 9999

APRENDER CON EL ORDENADOR

PARA LOS MAS JOVENES

■ Las horas

Este programa ha sido pensado para los más pequeños, sobre todo para aquellos que viven aferrados a los relojes digitales y olvi-

dan cómo leer una hora en un reloj analógico con manecillas.

Hemos creado dos versiones del programa, una para el Spectrum y otra para el PC y el MSX; la adaptación al AMSTRAD y el COMMODORE es sencilla y sólo requiere utilizar las rutinas correspondientes para el dibujo de la esfera del reloj.

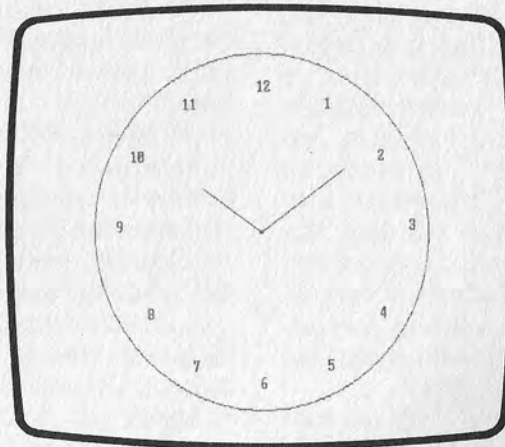
```
10 REM *****
20 REM ** PROGRAMA DE APRENDIZAJE DE LAS HORAS **
30 REM **          PARA ZX-SPECTRUM          **
40 REM *****
60 CLS
70 REM *****
80 REM ** ESFERA DEL RELOJ **
90 REM *****
100 LET C1=100:LET C2=80
110 CIRCLE C1,C2 ,75
120 CIRCLE C1,C2 ,2
130 REM *****
140 REM ** POSICIONES DE LA NUMERACION **
150 REM *****
160 DIM A(12),B(12)
170 FOR I=1 TO 12
180 READ A(I),B(I)
190 NEXT I
200 DIM TH(12,2),TM(12,2)
210 FOR I=1 TO 12
220 READ TH(I,1),TH(I,2),TM(I,1),TM(I,2)
230 NEXT I
240 REM *****
250 REM ** ESFERA CON NUMEROS **
260 REM *****
270 FOR I=1 TO 12
280 PRINT AT A(I),B(I):I
290 NEXT I
300 REM *****
310 REM ** GENERACION ALEATORIA DE LA HORA **
320 REM *****
330 LET H=INT(RND*12+1):LET H1=H
340 LET M=INT(RND*12+1):LET M1=M
350 IF M>6 THEN LET H=H+1
360 IF M=12 THEN LET H1=H1+1:M1=0
370 PLOT C1,C2 :DRAW TH(H,1),TH(H,2)
380 PLOT C1,C2 :DRAW TM(M,1),TM(M,2)
390 REM *****
400 REM ** PREGUNTA **
410 REM *****
420 INPUT "HORA ":HO
430 INPUT "MINUTO ":MI
440 IF HO=H1 AND MI=M1*5 THEN PRINT "CORRECTO":GOTO 460
450 PRINT "MAL"
460 PRINT "SON LAS ":H1:" ":M1*5
470 FOR I=1 TO 1000:NEXT I
480 RUN
1000 DATA 4,16,7,18,11,20,15,18,18,16,20,12,18,8,15,6,11,4,7,6,4,8,3,12
1010 DATA 15,40,25,50,30,30,40,40,40,0,60,0
1020 DATA 30,-30,40,-40,15,-40,25,-50,0,-40,0,-60
1030 DATA -15,-40,-25,-50,-30,-30,-40,-40,-40,0,-60,0
1040 DATA -30,30,-40,40,-15,40,-25,50,0,40,0,60
```

```
10 REM *****
20 REM ** PROGRAMA DE APRENDIZAJE DE LAS HORAS **
30 REM *****
40 RANDOMIZE TIMER
50 SCREEN 2
60 CLS
70 REM *****
80 REM ** ESFERA DEL RELOJ **
90 REM *****
```

```

100 LET C1=300:LET C2=150
110 CIRCLE(C1,C2),200,...,5/7
120 CIRCLE(C1,C2),2,...,5/7
130 REM *****
140 REM ** POSICIONES DE LA NUMERACION **
150 REM *****
160 DIM A(12),B(12)
170 FOR I=1 TO 12
180 READ A(I),B(I)
190 NEXT I
200 DIM TH(12,2),TM(12,2)
210 FOR I=1 TO 12
220 READ TH(I,1),TH(I,2),TM(I,1),TM(I,2)
230 NEXT I
240 REM *****
250 REM ** ESFERA CON NUMEROS **
260 REM *****
270 FOR I=1 TO 12
280 LOCATE A(I),B(I):PRINT I
290 NEXT I
300 REM *****
310 REM ** GENERACION ALEATORIA DE LA HORA **
320 REM *****
330 LET H=INT(RND*12+1):LET H1=H
340 LET M=INT(RND*12+1):LET M1=M
350 IF M>6 THEN LET H=H+1
360 IF M=12 THEN LET H1=H1+1:M1=0
370 LINE (C1,C2)-(TH(H,1),TH(H,2))
380 LINE (C1,C2)-(TM(M,1),TM(M,2))
390 REM *****
400 REM ** PREGUNTA **
410 REM *****
420 INPUT "HORA ":H0
430 INPUT "MINUTO ":M1
440 IF H0=H1 AND M1=M1*5 THEN PRINT "CORRECTO":GOTO 460
450 PRINT "MAL"
460 PRINT "SON LAS ":H1:" ":M1*5
470 FOR I=1 TO 1000:NEXT I
480 RUN
1000 DATA 4,42,7,49,11,53,16,49,19,42,20,33,19,24,16,18,11,14,7,16,4,23,3,33
1010 DATA 350,80,375,50,370,115,400,100,400,150,450,150
1020 DATA 370,185,400,200,350,220,375,250,300,250,300,250
1030 DATA 250,220,225,250,230,185,200,200,200,150,150,150
1040 DATA 230,115,200,100,250,80,225,50,300,50,300,10

```



PEQUEÑA HISTORIA DE LA INFORMATICA

■ Un avance fundamental: la tarjeta perforada



OMO bien sabemos, la tarjeta perforada nació en Europa, concretamente en Francia, donde Jacquard, basándose en otros trabajos anteriores (debidos a otro francés —Falcon—), la había utilizado con mucho éxito en su

telar. También sabemos que Babbage la integró en su máquina analítica (más bien en su proyecto de máquina analítica, porque nunca llegó a fabricarse). No sabremos nunca si esta máquina hubiera sido útil, o si no hubiera pasado de proyecto. En efecto, existen muchos tipos de personas creadoras: unas son «visionarias», saben adelantarse al tiempo, imaginar cosas originales, nunca pensadas por nadie, otras son fundamentalmente pragmáticas, y saben tomar pequeños rasgos, características e ideas de otros, y, sin grandes filosofías, llevarlas a la práctica obteniendo un producto útil, y en todo caso rentable. Finalmente, sólo unos pocos genios son capaces de unir esa imaginación de altura con dotes de sensatez: desde luego ellos son los verdaderos «genios», siendo los restantes (al menos para el que escribe) nada más y nada menos que «grandes hombres».

Babbage no fue un genio. Fue, sin embargo, un hombre excepcional. Hollerith tampoco lo fue, aunque sí perteneció al grupo de los pragmáticos capaces de ser útiles a la humanidad, y sacar provecho de sus cualidades.

¿Quién era Herman Hollerith? El «inventor» de la tarjeta perforada. Realmente, ya sabemos que no fue él quien la inventó, ya que

se había utilizado en los telares europeos y en los diseños de Babbage, pero sí fue el que la difundió, al menos para la resolución de problemas burocráticos que abrumaban a las empresas de la época.

Herman Hollerith tenía sólo diecinueve años en 1884, cuando pasó a formar parte como asistente del equipo del profesor William Trowbridge de la Universidad de Columbia. El profesor Trowbridge tenía relaciones en el Gobierno, y también en el Censo, de forma que a su asistente, Hollerith, le interesó el problema de clasificación de datos del Censo, del que hablaba su maestro. Tomó un empleo en la oficina, para mejor comprender los problemas que estaba investigando, y prosiguió con sus trabajos.

En el año 1886 la población de los Estados Unidos era, naturalmente, mucho menor que en nuestros días. Sin embargo, al menos para los funcionarios del Censo, eran «demasiado americanos». En efecto, por ese año la oficina debía confeccionar un nuevo Censo, y el problema acuciante era que no habían terminado aún de completar el Censo anterior. El agobio era enorme. El personal que trabajaba era muy numeroso, pero la cantidad de información era asfixiante. Y el Gobierno, por fin, dotó de suficientes ingresos a la oficina para que promoviera un concurso. El ganador de este concurso fue Herman Hollerith, que pasó a formar parte de la plantilla. Hollerith comprendió que la única solución posible era mecanizar el trabajo y organizar las tareas. Estudió las hojas del Censo y se dio cuenta de que muchas de las preguntas tenían respuestas sencillas, como, por ejemplo, sí, no o a veces tres o cuatro posibilidades como máximo. Dado el tipo de respuestas a procesar, se podrían utilizar tarjetas y perforar (o no perfo-

rar) determinadas posiciones. En cuanto a las contestaciones algo más complejas, siempre podrían representarse por grupos de perforaciones, creadas con un determinado código. Toda esta información codificada quedaba magníficamente almacenada en las tarjetas. Hemos de decir, sin embargo, que Hollerith comenzó a trabajar con tiras de papel, que tenían la ventaja de no desordenarse, etc. Sin embargo, el sistema era menos flexible, y Hollerith decidió pasar a utilizar tarjetas en lugar de papel continuo.

El siguiente paso fue un sistema eléctrico de detección de las perforaciones. Si existía una determinada perforación, se produciría un paso de corriente, y viceversa. El sistema consistía fundamentalmente en unas agujas que pasaban a través de las perforaciones, introduciéndose en un pequeño recipiente de mercurio, y completando así el circuito eléctrico. Si existía perforación, pasaba una señal eléctrica (uno lógico). Si no existía (en esa determinada posición de fila y columna) no pasaba ninguna corriente (cero lógico). Con todos estos elementos, y considerando que las operaciones fundamentales que llevaba a cabo la oficina eran de clasificación y recuento, Hollerith se puso en marcha, y creó una máquina que comenzó a trabajar con el Censo de 1890.

Realmente, el Censo de 1890 no fue lo primero que realizó la máquina de Hollerith. Previamente se había utilizado en la confección de unas tablas de mortandad en dos grandes ciudades americanas, Baltimore y Nueva York. Tampoco fue el Censo de 1890 el único trabajo que hizo Hollerith para esa oficina: su equipo recalculó el Censo de 1886 en un tiempo récord. Tardó dos años y medio en confeccionar un Censo de sesenta y dos millones de personas (imagínese el lector el número de tarjetas), cuando el tiempo requerido para realizar la operación (sin la ayuda de la máquina) superaba los siete años.

El éxito de estas máquinas fue enorme. Salieron fotografías en periódicos de la época e incluso en revistas científicas (*Scientific American*). Sin embargo, no todos fueron éxitos para Hollerith. Su patrón, el Estado americano, no estaba muy contento con su trabajo. ¿Por qué? Una costumbre medieval (no sabemos si muy práctica) hacía matar al mensajero de noticias no muy agradables. Y según los resultados del Censo, la población de los Estados Unidos no había crecido lo previsto,

¿Sabía usted que...

¿Sabe usted qué es una perforación de constantes o en serie?

Consiste en reproducir las perforaciones desde las tarjetas «maestras» a otras tarjetas llamadas «de detalle». Las tarjetas maestras llevan un indicativo especial para distinguirlas. Para reproducirlas basta con la unidad de perforación, conectando a los punzones de perforación las escobillas directamente, de forma que una tarjeta que se encuentre frente a dichas escobillas trasladará toda la información de sus perforaciones a la tarjeta siguiente, que en ese momento se encuentra justo frente a los punzones. Esta operación se repite hasta encontrar una tarjeta con el código indicativo de final de proceso, deteniéndose la operación.

También se pueden efectuar perforaciones de resumen o «sumarias», obteniendo la información de los registros de otra máquina en lugar de obtenerla de la forma que acabamos de indicar.

creando el consiguiente desasosiego en el Gobierno. Pero Hollerith era un triunfador. Volvió a su Albany natal y fundó una compañía, la Tabulating Machine Company, de la que se desprendería en 1911. Antes había pleiteado con el Gobierno, que se resistía a pagar los alquileres de sus máquinas. En 1910 el Gobierno decidió crear y utilizar sus propias máquinas para esta labor, poniendo al frente de las mismas a James Powers. Hollerith tuvo mucho éxito con su nueva compañía, que comerciali-



Herman Hollerith, inventor de máquinas automáticas de tabular.

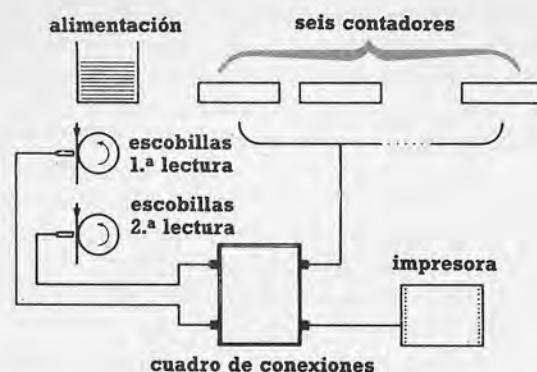
PEQUEÑA HISTORIA DE LA INFORMATICA

zó sus máquinas a nivel mundial. Y de esta compañía, fusionada con otras, se creó la International Business Machines Corporation (IBM) algunos años después. Hollerith ganó mucho dinero con sus negocios. Era un hombre muy práctico. Utilizó con el Gobierno el sistema de alquiler de las máquinas (frente a la venta), y además él también proporcionaba las cartulinas para las tarjetas. Hollerith no era hombre que no evolucionara y, como sabemos, no se contentó con las tareas típicas del Censo, de clasificación y recuento, pasando a mecanizar la mayoría de las tareas administrativas y aritméticas de oficina. Además, al cabo de pocos años la mayoría de los Censos de países industrializados utilizaban máquinas de Hollerith. El uso de las máquinas de tarjeta perforada tampoco se redujo únicamente a solucionar problemas típicos de oficina o comerciales. Poco tiempo después, estas máquinas eran utilizadas de forma eficaz en trabajos científicos. Sin embargo, para la resolución de muchos algoritmos, las máquinas fueron ineficaces. El problema fundamental era el tiempo requerido. Como siempre, si un problema como el Censo, o una predicción meteorológica, puede resolverse más tarde de un plazo razonable, ya no sirve para nada y, por tanto, pasa a no tener ningún interés la realización de los cálculos.

Hasta ese momento era de poco interés (o al menos eran poco fiables) afirmaciones hechas en los periódicos sobre las tendencias de la población (lo que hoy llamamos encuestas). Sin embargo, con la máquina de Hollerith se podían extraer datos interesantes. En octubre de 1890 el *New York Herald* publicó «Los campos a los que puede acceder la mujer trabajadora están ensanchándose muchísimo. Muchos empleos, que antes estaban cerrados para ella, ahora se están abriendo sin reservas... ¿En cuántos han probado ser más eficaces que los hombres? Al menos en el Censo han probado que lo son.»

Las máquinas de tarjeta perforada, por lo general, son máquinas independientes, es decir, realizan operaciones muy concretas. La información se perfora sobre tarjetas (semejantes a las que se han estado utilizando hasta hace pocos años), y puede procesarse o bien almacenarse.

La típica tarjeta conocida de todos está dividida en ochenta columnas y en doce filas (X, Y, 0, 1, 2, 3...). Con este sistema se pueden almacenar $12 \times 80 = 960$ dígitos binarios. Un 1 sería una perforación; un 0, la ausencia de di-



Tabuladora.

cha perforación. Los números decimales se registran escribiendo un dígito por columna, de forma que se pueden escribir números con ochenta decimales, o diez números de ocho dígitos, etc.

Los dígitos se escriben perforando las columnas adecuadas. Las líneas X e Y sirven para dar instrucciones especiales a la máquina o para indicar el signo de los números perforados en las otras diez líneas. También se utilizan para registrar información alfabética, por ejemplo, perforando la línea X y la línea 3, se habría anotado una C.

Hemos visto someramente cuál fue la solución de Hollerith a problemas de mecanización como el de la oficina del Censo. Sin embargo, es probable que no hayamos pensado que la máquina de Hollerith necesitaba además otra máquina que realizara las perforaciones de forma rápida y eficaz. (Considere el lector los errores que se podrían cometer si se perforara en la posición contigua a la deseada, por ejemplo.)

La máquina perforadora era muy semejante a una máquina de escribir (es decir, disponía de un teclado, y por cada golpe se pasaba a la siguiente posición de la tarjeta). Las perforaciones se llevaban a cabo por unos punzones especiales. Naturalmente, la velocidad (y fiabilidad) eran factores fundamentales. Operadores con experiencia podían conseguir perforar de 250 a 500 tarjetas en una hora.

La experimentación dio lugar a la aparición de otras máquinas que trataban las tarjetas: las verificadoras. Estas máquinas eran muy similares a las anteriores, pero en lugar de ir provistas de punzones llevaban unos percutores especiales. Las más comunes simple-

mente se detenían cuando no encontraban la perforación en su lugar.

La lectura de cada una de las tarjetas se realizaba en paralelo, es decir, las ochenta columnas a la vez. El equipo iba provisto de unas escobillas (ochenta) que hacían contacto con un rodillo metálico. Cuando aparecía una perforación, se producía una corriente eléctrica que era recogida por la escobilla correspondiente, que a su vez la enviaba a la unidad de interpretación de datos. El dato anotado es un valor lógico que corresponde al momento exacto en el que se producen las corrientes correspondientes (la fila) y las columnas que han permitido el paso de la corriente. Con esa información, la máquina responderá como haya sido preparada.

Veamos con algún detalle las tres máquinas básicas que componen un equipo de tarjeta perforada: la clasificadora, la reproductora y la tabuladora. La clasificadora selecciona las tarjetas leyendo una columna en cada pasada, colocándolas en unos cajetines numerados, correspondientes a los valores de esa columna. Con este sistema, si se va a clasificar un mazo de tarjetas por una numeración de tres dígitos, será necesario efectuar tres pasadas, si se clasifican por cinco dígitos cinco pasadas, etc. Terminada cada pasada, las tarjetas vuelven a reunirse en el orden adecuado, para volver a efectuar la pasada siguiente.

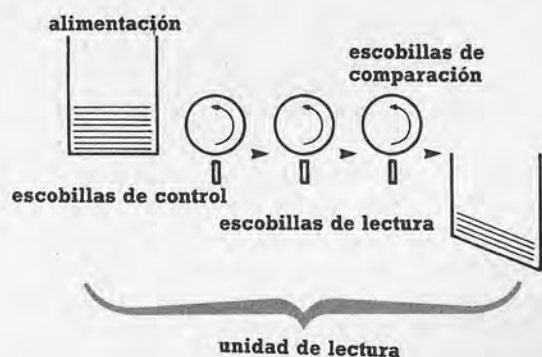
Las tarjetas van avanzando por un rodillo, con una escobilla situada en la columna que se desee. Pasan después por un mecanismo de selección que las envía a los cajetines correspondientes. El número de cajetines es igual al de las filas (doce) más uno para las tarjetas rechazadas. El mecanismo de clasificación consiste en un grupo de lengüetas de

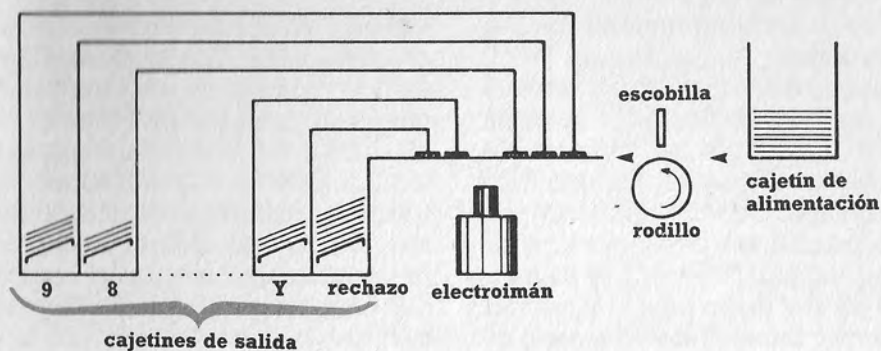
acero y unos electroimanes. Cuando éstos se activan, las lengüetas caen. Como esta operación se realiza en un determinado momento, es decir, cuando la tarjeta se encuentra sobre un determinado cajetín, el problema queda resuelto. Existen algunos tipos más modernos (para su momento, naturalmente) que contaban las tarjetas que existían en cada cajetín. Estas máquinas eran muy ingeniosas y útiles. También era «relativamente» rápidas, ya que podían clasificar de 450 a 800 tarjetas por hora, velocidad considerable en su momento.

La reproductora, otra de las piezas clave, estaba formada por dos partes muy independientes: la unidad de lectura y la de perforación. Cada una de estas unidades va provista de su propio alimentador de tarjetas, cajetín de salida, escobillas, etc. La unidad de perforación lleva, además, su juego de punzones. Estas dos zonas de la reproductora podían utilizarse de muchas formas, combinando las posibles conexiones de clavijas y conmutadores.

Las máquinas reproductoras podían «copiar» las perforaciones de un grupo de tarjetas en otro, tanto en las mismas columnas, como en otras distintas. Naturalmente, necesita dos juegos de tarjetas, unas vírgenes, que se colocaban en la unidad de perforación, y las maestras, que se colocaban en la unidad de lectura. Una vez realizada la copia, las tarjetas se verificaban para comprobar que las perforaciones estuvieran en los mismos lugares que las de las tarjetas maestras. Esta operación, muy importante, se realizaba mediante un juego de escobillas.

La tabuladora se encarga de realizar operaciones de lectura, operaciones aritméticas (de sumar y restar) y la impresión. Esta máquina lleva unos contadores (que son sim-





Clasificadora.

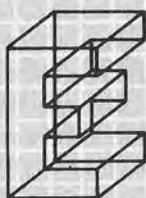
plemente unos registros acumuladores, donde los números se pueden sumar y restar de un registro a otro). La unidad de lectura dispone de un juego doble de escobillas, que lee y compara cada tarjeta con la siguiente, en un campo de columnas dado. Se utiliza principalmente para la creación de subtotales.

La unidad de impresión, dentro de la tabuladora, utiliza un sistema de impresión «en paralelo», es decir, imprime una línea completa cada vez (no va escribiendo secuencialmente cada letra, como ocurre en las máquinas de escribir). La información puede enviarse a

esta unidad de impresión tanto desde las propias tarjetas como desde los registros.

Pero no aburramos al lector con más detalles sobre las máquinas de tarjeta perforada. Aparecieron muchos otros modelos, que se fueron especializando para distintas tareas. También se extendieron ampliamente, y el ordenador no consiguió desterrarlas definitivamente hasta hace muy pocos años. Durante toda una época, el ordenador ha sido un elemento más (el más importante) dentro del equipo de mecanización de las empresas.

■ Primitivas en comunicación



N programación paralela uno de los temas más investigados es el desarrollo de las primitivas de comunicación. Las razones de que existan comunicaciones en programas en paralelo son las siguientes. En primer lugar,

se utilizan para enviar datos o señales a otro gránulo, también para sincronizar con otro gránulo, y, finalmente, para pedir que otro gránulo realice una determinada acción. Las comunicaciones se basan generalmente en el envío de mensajes o mediante el uso de variables compartidas. Como las variables compartidas se ha visto que no son útiles en programas que se ejecutan en una red de procesadores limitémonos a repasar las primitivas de comunicación basadas en el envío de mensajes, para recibirlos y para devolver respuestas. Las características importantes para estas primitivas de comunicación son los tipos de semántica de sincronización que pueden soportar, y si se ejecutan explícitamente o no. Las primitivas de comunicación pueden ser bloqueantes, o no bloqueantes. Se dice que una primitiva es bloqueante si el gránulo que está ejecutando esa primitiva es retardado hasta que se realice la comunicación. Si no es así, se llama no bloqueante. Si ambas primitivas (emisora y receptora) son bloqueantes, se dice que la comunicación obtenida es sincrónica. A diferencia del envío de mensajes, que siempre es explícito, la recepción puede ser a su vez explícita o implícita. En las recepciones explícitas, el proceso ejecuta expresamente una primitiva receptora para recibir un

mensaje, mientras que en la recepción implícita se activa una especie de procedimiento con la llegada del mensaje. Todas estas semánticas son útiles, ya que existen muchas aplicaciones diferentes que requieren distintos tipos de primitivas de comunicación. Un requisito que prevalece en las primitivas de comunicación a nivel de configuración es que su utilización debe ser transparente para las posiciones de los gránulos de comunicación, de forma que éstos puedan ser asignados a procesadores distintos, sin que se produzcan alteraciones. Los distintos sistemas disponibles utilizan combinaciones diferentes de primitivas.

Primitivas emisoras. Una propiedad importante en el envío de mensajes es qué ocurre si se retarda la emisión desde un gránulo. Se puede emitir por tres métodos: envío sin bloqueo, envío con bloqueo, y envío con llamada remota. El envío sin bloqueo hace que el proceso de emisión continúe su ejecución nada más invocar el comando de emisión. Concretamente, el gránulo emisor no es retardado incluso si el gránulo receptor no está preparado, por el momento, para la recepción del mensaje. La ventaja sustancial de esta primitiva es que se obtiene el mayor grado de paralelismo posible, ya que el emisor no espera jamás al receptor. El inconveniente de la emisión sin bloqueo es que la implementación necesita que los mensajes que se envían guarden cola, antes de ser recibidos. Como las memorias intermedias son finitas, el sistema operativo, o el soporte de tiempo de ejecución deben tomar las acciones oportunas en los casos en que no exista ya memoria disponible. La acción más común consiste en bloquear el gránulo que va a enviar el mensaje hasta que éste

pueda ser almacenado, lo cual está en tremenda contradicción con la semántica del envío sin bloqueo. Más aún, este tipo de bloqueos hacen que el programa paralelo quede estancado si todos los gránulos esperan que exista el consiguiente espacio disponible. Un método para obviar estos estancamientos podría ser limitar el número de mensajes que envía cada gránulo, para evitar que se produzcan mensajes con demasiada rapidez. (Esta información puede, evidentemente, especificarse dentro de la configuración.) Por otro lado, el sistema puede desechar el mensaje más antiguo (o el más moderno), en lugar de bloquear el envío desde el gránulo. El mantener solamente el último mensaje (el más moderno) es la operación más sensata para sistemas de control de tiempo real, si los gránulos sólo necesitan el estado actual de los dispositivos controlados. Pero si se desecha el último mensaje, el más moderno, la operación sólo tendría sentido en el supuesto de que el que lo envía puede detectar que el mensaje ha sido eliminado, y, por consiguiente, puede enviar de nuevo el mensaje.

La naturaleza asíncrona de las emisiones sin bloqueo causa dificultades de programación en casos de situaciones excepcionales y verificaciones de la corrección. Si ocurre un error durante la transmisión del mensaje (como, por ejemplo, la terminación del gránulo receptor o un mal funcionamiento del medio de comunicación), es difícil devolver un código de error al emisor, ya que la ejecución ha seguido arbitrariamente desde el punto en que se realizó el envío, o también el propio emisor ha podido terminar cuando ocurrió la situación excepcional. Por estas razones, los programas que utilizan comunicaciones asíncronas son mucho más complicados de verificar que los que utilizan comunicaciones síncronas. En el primer caso, el estado actual del emisor puede no corresponderse con el estado en el que se envió el mensaje, y, por tanto, cuando éste se reciba, el receptor no podrá hacer aseveraciones sobre el estado del emisor.

En emisiones con bloqueo, los gránulos que envían mensajes se retardan hasta que el gránulo receptor pueda recibirlos. Como el gránulo emisor está bloqueado, no es necesario utilizar memorias intermedias para mensajes en la implementación. Además, tampoco serán necesarias primitivas de sincronización adicionales, ya que el emisor y el receptor están sincronizados desde el momento en que se

comienza la comunicación. Los programas que están escritos utilizando emisiones con bloqueo son muy fáciles de verificar, ya que la instrucción de envío con bloqueo y su correspondiente instrucción receptora se pueden considerar como una única instrucción de asignación que está «distribuida» (en dos módulos de programa distintos). Pero no pensemos que este tipo de envíos no tiene inconvenientes de ningún tipo. El principal, posiblemente, es que al utilizar envíos con bloqueo se inhibe el paralelismo, ya que el gránulo emisor no puede seguir su ejecución hasta que el gránulo de destino esté preparado para recibir su mensaje. Otra de las desventajas es que la implementación del envío con bloqueo requiere dos conmutaciones adicionales, una para ejecutar otro proceso, mientras el emisor espera que se reciba el mensaje, y otro para continuar el envío.

La tercera posibilidad es el envío por llamada remota. Con este sistema, el gránulo emisor espera hasta que recibe una respuesta explícita del gránulo receptor. Después de recibir un mensaje el gránulo receptor ejecuta un procedimiento para devolver la contestación. Esta primitiva es análoga a la llamada de un procedimiento en los lenguajes secuenciales. Esta forma de comunicación es clara y además eficaz cuando se corresponde con la estructura de comunicación de los programas de aplicación. Sin embargo, algunos patrones de comunicación que son muy útiles no pueden programarse usando esta forma. Entre ellos citamos comunicaciones que no necesitan respuesta por cada petición, o cuya respuesta debe llegar desde otro proceso distinto del receptor original. Es importante fijarse en que el envío bloqueado es un caso especial de llamada remota, sin contestación.

Veamos ahora las primitivas que deben contestar.

En muchos lenguajes y sistemas que soportan envíos por llamadas remotas, el proceso objeto de la respuesta es implícito, ya que debe haberse recibido el mensaje de este proceso para que la contestación sea válida. Así se ve reflejada la analogía entre los envíos por llamada remota y las llamadas a un procedimiento. La utilización de nombres implícitos, sin embargo, requiere que la respuesta al mensaje sea enviada antes del siguiente mensaje, o que los mensajes recibidos sean apilados y sacados de la pila según se vayan generando las respuestas. El lenguaje ADA, por ejemplo, utiliza este procedimiento de alma-

cenar mensajes en una pila. La posibilidad de contestar mensajes en un orden cualquiera (no relacionado con el orden en el que los mensajes se reciben) añade mucha potencia y versatilidad al envío por llamada remota. Esto último puede conseguirse especificando el nombre del proceso objetivo, explícitamente dentro del comando respuesta. Los procesos dependientes de otros, que necesitan manejar los recursos internamente, pueden codificarse con frecuencia bastante eficazmente mediante respuestas explícitas. Sin embargo, el inconveniente es que la estructura del programa se complica enormemente, haciendo muy difícil su comprensión.

En lo que respecta a las primitivas receptoras, la recepción de mensajes se puede considerar activa o pasiva, dependiendo de si el proceso ejecuta explícitamente operaciones de recepción. Los lenguajes que soportan recepción activa ofrecen operaciones de recepción que pueden ser ejecutadas explícitamente por los gránulos. La forma más común de estas «operaciones» de «recepción activa» bloquea el gránulo receptor hasta que llegue el mensaje, ya que el proceso receptor no tiene otra tarea que hacer mientras espera el mensaje. Por otro lado, está la recepción «sin bloqueo», que nunca produce retardos en el gránulo que se recibe. Entre ambas posibilidades está la recepción de bloqueada, con «timeout» (control de límite de tiempo), que limita el tiempo que el gránulo debe esperar a que llegue el mensaje.

Una señal de recepción «con bloqueo» realiza una sincronización implícita del módulo (gránulo) receptor, desde que dicho módulo receptor se retarda hasta después del envío del mensaje. Si se utiliza una recepción sin bloqueo, será necesario utilizar algún procedimiento de espera-ocupado, para resolver el problema de la sincronización. En los casos en los que un único gránulo se deba comunicar con muchos otros, la dificultad de usar recepción con bloqueo estriba en que debe determinarse a priori el orden en el que se van a establecer las comunicaciones. Para permitir recepciones no determinísticas desde varios gránulos, existen muchos sistemas y lenguajes que soportan diferentes formas del comando de control, y que se pueden utilizar para esperar los mensajes en el caso de que éstos provengan de distintos gránulos. También se pueden obtener recepciones no determinísticas utilizando técnicas de recepción condicionales. En estos casos, las instrucciones recep-

toras incluyen expresiones Booleanas, que se utilizan para seleccionar el mensaje deseado entre varios mensajes pendientes. Las recepciones sin bloqueo también se pueden utilizar para recibir mensajes de forma no determinística, mediante un sistema de llamadas alternativas. En recepciones pasivas, la llegada de un mensaje comienza la ejecución del correspondiente cuerpo de código. Dicho con otras palabras, el receptor proporciona un conjunto de manejadores de mensajes, y, se llamará al manejador adecuado en cuanto llegue el mensaje al receptor. Entre los manejadores de señales, podemos indicar el soporte en UNIX, ciertas primitivas de comunicación utilizadas en DP y Argus, y las llamadas a procedimientos remotos implementadas en Xerox PARC. La ventaja mayor de las recepciones pasivas consiste en que el orden en el que se manejan los mensajes está determinado por los emisores, en lugar de serlo por el receptor. Este punto es importante, al ayudar a explotar mucho más el paralelismo, ya que los procesos se ejecutan dependiendo del controlador que actúa bajo demanda en lugar de hacerlo de forma predeterminada. El proceso de recepción puede manejar varios mensajes, de forma concurrente o secuencial, dependiendo de la implementación, y semántica. La recepción pasiva está asociada generalmente con la emisión de llamadas remotas. Así pues, para obtener el máximo grado de paralelismo posible, se pueden utilizar las emisiones sin bloqueo con recepción pasiva.

Vamos a definir ahora un nuevo término. La conectividad. Este término indica el número de gránulos que comparten el mismo canal, en un momento dado. Existen nueve opciones posibles: uno con uno, uno con muchos, uno con cualquiera, cualquiera con uno, cualquiera con muchos, cualquiera con cualquiera, muchos con uno, muchos con muchos, y muchos con cualquiera. La primera opción, uno con uno, significa que el canal de comunicación conectará un emisor con un receptor. Por tanto, utilizando este tipo de canal, sólo un único mensaje emisor podrá enviar un mensaje al receptor. Las posibilidades cualquiera con uno, y muchos con uno significan que se pueden tener varios emisores conectados con un receptor. Si el receptor puede recibir un mensaje desde cualquier emisor, la emisión será de cualquiera con uno. La emisión de muchos con uno significa que se recibe el mensaje cuando todos los emisores han enviado mensajes. El PCL puede soportar este

tipo de envíos, pero su utilidad no ha quedado muy clara, por el momento. Las emisiones uno con muchos, y uno con cualquiera, significan que un emisor está conectado a varios receptores. Cuando cada mensaje es enviado por el emisor a todos los receptores, se trata de una emisión uno con muchos. Se ha demostrado con la experiencia, que la emisión de mensajes multidestino es mucho más eficaz que varias transmisiones uno con uno, incluso en los casos en los que la red local de la que hablamos no soporta la difusión (broadcasting). La emisión uno con cualquiera permite al emisor enviar un mensaje a un receptor concreto, en lugar de hacerlo a todos los receptores. Esto último puede emularse utilizando varias conexiones uno a uno, sólo si el emisor conoce a priori cuál de los gránulos debe recibir el mensaje. Las conexiones muchos con muchos combinan los tipos muchos con uno y uno con muchos. Aunque es posible emular las conexiones cualquiera con uno y uno con muchos mediante conexiones uno con uno, el sistema resultante no es en absoluto práctico. El sistema MODEL, por ejemplo puede soportar las conexiones uno con uno, cualquiera con uno, y uno con cualquiera.

Los formatos permitidos para los mensajes pueden ser de dos tipos: en formato no imprimible o en formato imprimible. Los mensajes «no imprimibles» son mensajes que se consideran como una secuencia de bytes, cuyo significado está establecido entre el emisor y el receptor. Los sistemas y lenguajes que utilizan este tipo de mensajes suelen limitar la longitud de los mensajes a un valor fijo, aunque también pueden permitir, en ciertos casos, mensajes de longitudes variables. Los mensajes no imprimibles son más sencillos de manejar y más eficaces que los imprimibles, por lo que es frecuente que se utilicen en la implementación. Sin embargo, los programas escritos utilizando mensajes no imprimibles suelen presentar problemas a la hora de su depuración, siendo mucho más sencillos de depurar aquellos que utilizan mensajes imprimibles. De esto último se deduce que los sistemas de comunicación que utilicen programas de aplicación, deben soportar mensajes imprimibles.

Es útil comprobar que, para cada mensaje enviado y recibido, es importante que el tipo del mensaje emitido sea idéntico al tipo de mensaje que espera el receptor. Así, la comunicación puede comprobarse de dos formas: bien estáticamente, o dinámicamente.

La ventaja de la comprobación estática de consistencia, es que se detectan inmediatamente las faltas de concordancia en el tipo. Las comprobaciones dinámicas de consistencia aportan la ventaja de que pueden utilizar el mismo canal de comunicación tanto para emitir como para recibir mensajes con representaciones diferentes. Una de las desventajas de la comprobación dinámica de la consistencia es que no puede detectarse si la comunicación es inconsistente hasta el momento de la ejecución, lo que significa que todas las excepciones producidas por error en el tipo de mensaje deben manejarse y valorarse durante la ejecución. Otra de las desventajas es que existe un overhead (proceso adicional que carga el sistema) durante la ejecución, para comprobar la consistencia del mensaje durante ésta.

Además, la información, sobre el tipo de mensaje necesita ser enviada con datos, y, por tanto, deben transmitirse más bits. Este overhead que se produce en tiempo de ejecución, se puede disminuir utilizando comprobaciones dinámicas de consistencia, basadas en mecanismos del tipo «función hash». Las comprobaciones de tipo se realizan utilizando valores «hash» de las descripciones de tipo, calculados en el momento de la compilación, y el tamaño del «rango» de los valores utilizados por la función hash determina el grado de seguridad obtenido.

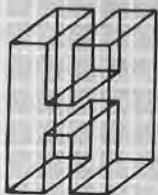
Se podría establecer cuáles son los tipos de mensajes permitidos basándonos en el tema; o del overhead producido antes y después de la transmisión del mensaje. Si no se permiten indicadores dentro del mensaje, se produce el overhead, tanto si es de tamaño fijo o variable, al copiar mensajes en y desde las memorias de transmisión. Sin embargo, si se permiten indicadores dentro de los mensajes, éstos podrán ser estructuras de datos arbitrariamente complejas. En estos casos, las estructuras de los datos son transformadas antes de ser copiadas en las memorias intermedias de transmisión, convirtiéndose los indicadores en direcciones interiores de esas memorias antes de la transmisión. Estas estructuras de datos se reconstruyen nada más llegar a la recepción. También es necesario preservar la partición y la rotación cíclica dentro de las estructuras de datos transmitidas, y esto puede hacerse utilizando durante las conversiones una tabla de traducción entre indicadores y direcciones. Dichas conversiones deben ser trans-



parentes para el programador, y se podrían realizar por el sistema automáticamente durante la comunicación. La transmisión de mensajes se hace mucho más compleja si los mensajes contienen tipos de datos abstractos, como, por ejemplo, una pila (stack) o una tabla de símbolos. Si diferentes implementaciones utilizan representaciones diferentes, para el valor del dato abstracto, como resultado se produce un overhead, obtenido al traducir un valor abstracto almacenado en una representa-

ción dentro de otra representación, durante la transmisión. Existe un método para transmitir estos tipos de datos abstractos. Sin embargo, el método requiere que el programador escriba las operaciones de codificación y decodificación para cada implementación de un tipo de dato abstracto. Los valores abstractos se codifican en representación canónica, antes de la transmisión, para, a continuación, decodificarlos en la representación interna del receptor.

■ REPERTORIO DE LENGUAJES DE CONFIGURACION



AGAMOS un resumen breve de algunos lenguajes de configuración. La mayoría de los lenguajes de programación paralela, orientados a mensajes incluyen características que en cierto modo deben considerarse como lenguajes de configuración. Por ejemplo, en la mayoría de los lenguajes de programación, las relaciones interprocesales se expresan dentro de cada proceso individual. Esta aproximación es únicamente para los lenguajes desarrollados específicamente para describir especificaciones de configuración separadas. Veamos algunos de ellos.

NETSLA. Este lenguaje orientado a mensajes es útil para separar las relaciones interprocesales a partir de las descripciones de los procesos. Consiste esencialmente en dos sublenguajes, un lenguaje de especificación de la red local, llamado NETSLA, y un lenguaje de descripción del proceso, llamado ALSTEN. Los procesos están escritos en ALSTEN, que es una extensión del PASCAL que permite la declaración de puertas que permitan soportando operaciones de transmisión de mensajes. Estas declaraciones están ya separadas de las definiciones propias de la red, y puede considerarse que forman una unidad. A su vez, ALSTEN se ocupa básicamente de la emisión sin bloqueo, ya que el envío se bloquea hasta que el valor del mensaje se copie en la memoria intermedia de mensajes de los receptores, que deben tener una capacidad de recepción no determinística. Permite conexiones del tipo cualquiera con uno y uno con muchos. NETSLA tiene capacidad para que el programador especifique la configuración inicial, creando los procesos y definiendo las conexiones entre puertas. La consistencia de las conexiones se basa en procedimientos de comprobación del PASCAL, muy fiables. El manejador de sucesos puede modificar la

topología de la red creando procesos, borrándolos y redefiniendo las conexiones entre puertas.

PCL. Siglas de Process Control Language (Lenguaje de Control de Procesos). Es un lenguaje, como indica su nombre, pensado para el control de procesos y diseñado especialmente para poder especificar estructuras de procesos muy largas y complejas, sobre redes locales de procesadores. En el momento de su creación, apenas si existían lenguajes que solucionaran los problemas de las especificaciones de este tipo de procesos largos y complejos. Su estructura básica está formada por «celdas», que son grupos de componentes como pueden ser procesos, segmentos de memoria (espacios compartidos), puertas y uniones (links) entre módulos. Cada una de estas celdas puede a su vez contener subceldas, creando una descripción jerárquica de la estructura del proceso. La estructura del proceso se determina de un modo estático, sin embargo, las instancias del proceso y los segmentos de memoria se crean en el momento de la ejecución.

La descripción de la estructura de un proceso está formada fundamentalmente por tres partes: parte estática, parte evolutiva y creación. La parte estática identifica los componentes de la red del proceso y sus posibles topologías. La parte evolutiva describe el momento de la creación de los componentes, y cómo se crearon, ejecutaron, eliminaron, etc. La última parte, de creación, especifica hasta qué punto se comparten copias de unos componentes por parte de otros componentes de la estructura, es decir, cómo cambia la topología al crearse nuevos componentes en el momento de la ejecución. Como los segmentos de memoria, etc., se crean en el momento de la ejecución, sus conexiones se efectúan según las necesidades. Para poder disponer de estas uniones dinámicas, y conexiones, el lenguaje dispone de nueve tipos distintos de

uniones (links), con tres posibles emisores y tres posibles receptores. Este lenguaje tiene todavía que desarrollarse, sobre todo en lo que respecta a la depuración y localización de recursos.

CONIC. Consiste en un conjunto de herramientas integrado para la construcción y manejo de sistemas de control de conjuntos distribuidos de ordenadores. Fundamentalmente, se detiene en los conceptos modulares de software para la construcción de programas, y en la creación de sistemas con módulos individuales de software, distribuidos en una determinada configuración.

La unidad principal es el módulo, que está formado por puertas, tipos, variables, procedimientos, tareas, etc. Las puertas son módulos de interface que permiten la interconexión de otros módulos de muy diversas formas. Las primitivas de comunicación son del tipo emisión sin bloqueo y recepción con bloqueo. El tamaño de la cola de las memorias intermedias para cada puerta se determina estáticamente (a priori). Cuando se sobrepasa el espacio disponible, se sobrescribe el mensaje más antiguo. Las interconexiones entre gránulos pueden ser de tipo uno con uno, cualquiera con uno y uno con muchos.

La configuración del sistema consiste en la definición del tipo, creaciones de módulos e interconexiones. En las últimas versiones, el CONIC puede soportar modificaciones de la configuración del sistema permitiendo creación dinámica de módulos (por supuesto borrado, interconexión, etc).

DICON. Ha sido desarrollado para la creación de programas paralelo en tiempo real, fundamentalmente para sistemas multisensores. Los programas paralelo en tiempo real están contruidos configurando un conjunto de programas secuenciales, utilizando un lenguaje de configuración distribuido llamado DICON. Los gránulos del sistema se pueden programar en varios lenguajes, como LISP, C o PROLOG, dependiendo de la tarea que realicen, para que sean más eficaces. El lenguaje DICON se ha diseñado especialmente para que soporte la construcción modular de los programas paralelos, y que pueda expresar la información necesaria para crear, cargar y ejecutar un programa paralelo. La configuración de un programa paralelo consiste en la declaración del proceso, los requerimientos, las

restricciones a tener en cuenta en el proceso en tiempo real, prioridades y otras instrucciones de control del proceso. El programa paralelo se construye compilando la configuración, y se carga y ejecuta, solo, o bajo la supervisión de un intérprete de comandos, que permite al programador seguir el proceso y controlar la ejecución de cada gránulo, para evitar posibles problemas.

El lenguaje soporta comunicaciones unidireccionales y bidireccionales. Los gránulos emiten mensajes con un sistema sin bloqueo, utilizando una recepción con bloqueo.

El programador construye el programa distribuido y lo ejecuta. El método es el siguiente: primero crea los subsistemas y programas componentes, después describe la configuración de los programas paralelo en DICON, luego compila la configuración, y finalmente la ejecuta, pudiendo seguir y controlar la ejecución del proceso de forma interactiva, si lo desea.

CSL. Siglas de Computation Structures Language. Fue desarrollado específicamente para el ordenador matricial de Texas Instruments, TRAC (Texas Reconfigurable Array Computer). Su labor fundamental es separar la estructuración paralela de la secuencial, en los programas. Utiliza módulos escritos en lenguajes de alto nivel para los módulos de programas paralelo, y proporciona comunicaciones con memoria compartida, basadas en el envío de mensajes. También permite la creación de canales de comunicación entre los elementos. El programa consiste esencialmente en secuencias de instrucciones de creación, y otras de control de la ejecución.

Las instrucciones de creación especifican cada nueva topología, indicando las tareas, datos compartidos y canales lógicos. Estas especificaciones se utilizan para configurar una arquitectura que sea conforme con la topología, y con el mapa de tareas y canales de dicha arquitectura. Las instrucciones de control de la ejecución definen el momento en el que se ejecutarán las tareas, qué tipo de datos compartidos se usarán y cuándo se llevarán a cabo las comunicaciones. Las primitivas de comunicación son de envío sin bloqueo, con una memoria de tamaño fijo y una recepción con bloqueo. La transmisión de mensajes que pueden soportar es de tipo uno con muchos.

VOCABULARIO DE INFORMATICA

Label. Término inglés para designar una etiqueta. (Véase **Etiqueta.**)

Lápiz óptico. Dispositivo de comunicación entre el usuario y el ordenador. Su aspecto es muy parecido al de un lápiz normal. Se suele utilizar en terminales de ordenador para referenciar puntos en la pantalla. El dispositivo es controlado por el programa correspondiente.

Si se considera el lápiz óptico como periférico de entrada, introduce informaciones de un bit, interpretadas por el programa y utilizadas por el hardware adecuado.

Latencia. Tiempo transcurrido entre la terminación completa de la interpretación de una dirección y la transferencia desde esa posición indicada. En esta latencia está incluido el retardo asociado con el acceso a los dispositivos de almacenamiento, etc.

Layout. (Véase **Presentación.**)

Leading. Término inglés utilizado para designar cualquier elemento situado al comienzo de algo. Puede referirse a ceros (leading zeros) que son los ceros situados a la izquierda de un determinado número, leading edge, flanco inicial, etc.

Learning. Término inglés para designar un aprendizaje de cualquier disciplina. En informática suele aplicarse a enseñanzas asistidas por ordenador.

Least significant bit. Término inglés para designar el bit menos significativo, es decir, aquel cuyo peso es menor. Generalmente es el bit situado más a la derecha, pero existen casos en que esta regla no se cumple.

Least significant digit. Término inglés para designar el dígito menos significativo, o aquel cuyo peso es menor. Suele ser el situado más a la derecha.

Lectora. Dispositivo capaz de leer un soporte de memoria externo. Existen muchos tipos de lectoras, dependiendo del tipo de dispositivo que sean capaces de leer, como, por ejemplo, lectora de cintas, de tarjetas, etcétera.

Lectura. Operación por la que se toman o interpretan datos que se encuentran en un dispositivo de almacenamiento, u otro medio o fuente. La operación inversa se llama escritura, o grabación.

Left. Término inglés que significa izquierda. Puede aplicarse a casi cualquier cosa, por ejemplo, «left justify», ajuste por la izquierda, etc.

Length. Término inglés para indicar la longitud. Puede referirse a bloques, registros, palabras, etc.

Lenguaje. Conjunto de caracteres y reglas con las que se puede escribir un programa de ordenador para manejar adecuadamente la información. Existen muchos tipos de lenguajes, aquellos que utilizamos para referirnos a otros, llamados metalenguajes, lenguajes naturales (en contraposición a los artificiales), lenguajes orientados al objeto, lenguajes orientados al procedimiento, lenguajes fuente, lenguajes de programación, etcétera.

Lenguaje ensamblador. Lenguaje de programación muy similar al lenguaje máquina y, por tanto, distinto para cada ordenador. Es un lenguaje simbólico, es decir, a cada instrucción le corresponde una única instrucción de máquina. Estos lenguajes próximos al lenguaje máquina son muy incómodos de utilizar, pero aprovechan al máximo los recursos y posibilidades de cada ordenador.

